

Quantitative classical realizability

Aloïs Brunel^a

^a*LIPN - UMR CNRS 7030 - Université Paris 13, Villetaneuse, France*

Abstract

Introduced by Dal Lago and Hofmann, quantitative realizability is a technique used to define models for logics based on Multiplicative Linear Logic. A particularity is that functions are interpreted as bounded time computable functions. It has been used to give new and uniform proofs of soundness of several type systems with respect to certain time complexity classes. We propose a reformulation of their ideas in the setting of Krivine's classical realizability. The framework obtained generalizes Dal Lago and Hofmann's realizability, and reveals deep connections between quantitative realizability and a linear variant of Cohen's forcing.

1. Introduction

Ever since its introduction by J.L Krivine [12], the theory of classical realizability has raised a growing interest. Initially designed to study the computational content of classical proofs through the Curry-Howard correspondence, it has led to promising results in various fields. One could mention the recent advances [15] made by Krivine in the elaboration of new models of the **ZF** axiomatic set theory. Another success has been its use to define and justify a classical extraction procedure for the proof assistant Coq [21].

Forcing — Forcing is a technique designed by Cohen [4] to prove the independence of the Continuum Hypothesis (CH) from **ZFC**. The idea is to define a formula transformation which turns every formula A into a new one noted $p \Vdash A$, where p is a *forcing condition*. By choosing a suitable set of forcing conditions, one can prove the statement $p \Vdash \neg CH$. It has been recently shown by Krivine [14] that combining classical realizability and forcing is possible. This construction can be seen as a generalization of forcing iteration and makes possible a study of forcing through the Curry-Howard isomorphism: Krivine has shown that the forcing technique not only provides a logical translation but also a program transformation. Following that work, Miquel [22] has introduced an abstract machine (the Krivine Forcing Abstract Machine, or **KFAM**) that

Email address: alois.brunel@ens-lyon.org (Aloïs Brunel)

internalizes the computational behavior of programs obtained via this transformation. One remarkable feature of this machine is that it provides sophisticated programming features like memory cells or program execution tracing.

Resource sensitive realizability — Realizability techniques have also been fruitfully applied to *implicit complexity*. This research field aims at providing machine-independent characterizations of complexity classes (such as polynomial time or logspace functions). One of the possible approaches is to use linear logic based type systems to constrain programs enough so that they enjoy bounded-time normalization properties. Proving these properties can be achieved using semantic techniques. Following different works [9, 10], Dal Lago and Hofmann have introduced in [18] a *quantitative* (another word for *resource sensitive*) framework based on Kleene realizability [11]. One of the crucial ideas behind Dal Lago and Hofmann’s work is to consider bounded-time λ -terms as realizers. Bounds are described using elements of a *resource monoid*. No matter what resource monoid is chosen, their framework always yields a model of second-order Multiplicative Affine Logic (**MAL**). Various systems extending **MAL** are then dealt with by choosing a suitable resource monoid, while the basic realizability constructions are unchanged. This work has offered new and uniform proofs of the soundness theorems for **LAL**, **EAL**, **SAL** and **BLL** with respect to the associated complexity classes [16, 17, 18]. In [3], Terui and the author gave a new characterization of the complexity class **FP** (the functions computable in polynomial time) and used a variant of Dal Lago and Hofmann’s realizability to show the soundness part of this result.

The present work aims at applying methodology and tools coming from classical realizability to generalize the framework proposed by Dal Lago and Hofmann, and to reveal deep connections between quantitative realizability and forcing techniques.

Quantitative classical realizability — We propose a new quantitative framework, based on Munch’s classical realizability for focalising system **L** (or **L_{foc}**) [23], a term calculus for classical logic **LC** [8]. We extend this realizability using the notion of *quantitative monoid*, which derives from the *resource monoid* structure introduced by Dal Lago and Hofmann. We show that, whatever the quantitative monoid, this framework always gives rise to a model of the Multiplicative Affine fragment of Higher-order Classical Arithmetic (abbreviated **MAL ω**). By choosing different quantitative monoids, we obtain models of logics extending **MAL ω** . Because all resource monoids in the sense of [18] are also quantitative monoids, we can in principle obtain models for all the systems treated in [17, 18], although we only exhibit a model of Soft Affine Logic (**SAL**) [2].

Quantitative reducibility candidates — By carefully setting parameters of classical realizability, one can retrieve the notion of *reducibility candidates* (presented using orthogonality, as in [7, 19, 24, 25]), which is used to

prove normalization properties. Similarly, in our setting, we are able to define a quantitative extension of this technique, which we call *quantitative reducibility candidates*. It allows us to semantically prove complexity properties of programs that are typable in the logic we interpret. Moreover, because we work with a term calculus which generalizes both call-by-name and call-by-value classical λ -calculi, these complexity properties are transferred for free to these calculi. Hence, we are able to retrieve and generalize the bounded-time termination results proved in [17, 18].

A forcing decomposition — Quantitative classical realizability is deeply connected with a certain notion of forcing, which we propose to study. We formalize inside $\mathbf{MAL}\omega$ a forcing transformation on Multiplicative Linear Logic (\mathbf{MAL}) formulas, called *linear forcing*. Then, following Miquel’s methodology [22], we propose an abstract machine designed to execute programs obtained by a specific linear forcing instance. A **connection lemma** is proved, which shows that composing this instance of linear forcing with a non-quantitative realizability built upon this machine always yields a quantitative realizability model. Finally, using this result, we show how quantitative reducibility candidates (restricted to \mathbf{MAL}) arise from the composition of usual reducibility candidates with forcing.

Outline — Sections 2 and 3 introduce $\mathbf{MAL}\omega$ and its quantitative realizability interpretation. The model of quantitative reducibility candidates is then defined and used to prove a bounded time termination property of $\mathbf{MAL}\omega$. In section 4, we show by taking \mathbf{SAL} as an example that this interpretation and the corresponding complexity result can be extended to larger type systems. Finally, we introduce in section 5 the linear forcing interpretation of \mathbf{MAL} and prove the accompanying decomposition results.

2. The calculus

In this section, we describe the system $\mathbf{MAL}\omega$. It is based on the \mathbf{MAL} type system for Munch’s focalising system \mathbf{L} [23], extended with higher-order quantifications and arithmetical operations. Logically, it is a fragment of classical higher-order Peano arithmetic (abbreviated by $\mathbf{PA}\omega$). The syntax of $\mathbf{MAL}\omega$ is divided in three distinct layers: the *terms*, the *type constructors* and the *kinds*. The language of terms, which we shall use to express both proof-terms and realizers, is based on the multiplicative fragment of \mathbf{L}_{foc} , extended with extra instructions. The type constructors layer is an adaptation of the higher-order terms syntax of $\mathbf{PA}\omega$ [22] to linear logic: it can be seen as a combination of the languages of $\mathbf{PA}\omega$ and classical $\mathbf{F}\omega$ [19]. Finally, kinds are used as a simple type system for type constructors.

2.1. Term syntax

In what follows, *positive variables* and *negative variables* are respectively written x, y, z, \dots and $\alpha, \beta, \gamma, \dots$. We use the symbols κ, κ', \dots to denote both

positive and negative variables. In the term syntax of \mathbf{L}_{foc} , in addition to variables, six syntactic categories are defined: *values*, *positive values*, *positive terms*, *negative terms*, *terms* and *commands*:

variables	κ, κ'	$::=$	$\alpha \mid x$	
values	V	$::=$	$V_+ \mid t_-$	
positive values	V_+	$::=$	$x \mid (V, V') \mid \{V\} \mid k_+$	$(k_+ \in \mathcal{K}_+)$
positive terms	t_+	$::=$	$V_+ \mid \mu\alpha.c$	
negative terms	t_-	$::=$	$\alpha \mid \mu(\kappa, \kappa').c \mid \mu\{\kappa\}.c \mid \mu x.c \mid k_-$	$(k_- \in \mathcal{K}_-)$
terms	t, u	$::=$	$t_- \mid t_+$	
commands	c	$::=$	$\langle t_+ \mid t_- \rangle$	

where $\mu(\kappa, \kappa').c$ is not defined if $\kappa = \kappa'$. Moreover terms are always considered modulo α -equivalence. We also make an identification between the commands $\langle t \mid u \rangle$ and $\langle u \mid t \rangle$. Finally, we associate to every term t its *polarity* $\pi(t) \in \{-, +\}$ as follows:

$$\pi(t) = \begin{cases} + & \text{if } t \text{ is a positive term} \\ - & \text{if } t \text{ is a negative term} \end{cases}$$

Remarks 1.

1. Notice that in the definition of the pair construct (V, V') , V and V' can be values of arbitrary polarity (that is, positive or negative). We could have made the choice of restricting a pair to positive values. This would not be problematic since we still could use $\{.\}$ to change the polarity of values from negative to positive before putting them into a pair.
2. The term $\{V\}$ can be seen as a one-tuple and is here to give the possibility of turning a negative term into a positive one.
3. This untyped calculus has no linear restriction on the use of variables. However, such restrictions will appear in the type system.
4. The identification of $\langle t \mid u \rangle$ and $\langle u \mid t \rangle$ accounts for the involutivity of linear negation.

Similarly to [22], the syntax is parametrized by a set of *positive instructions* \mathcal{K}_+ (which are considered as values) and a set of *negative instructions* \mathcal{K}_- . This allows us to extend the language at will, in the spirit of Krivine's λ_c -calculus [12].

Remark 2. If we want to make a comparison with Miquel's work [22], our set \mathcal{K}_- corresponds to the set of *instructions* while \mathcal{K}_+ corresponds to the set of stack constants.

If x is a term or a command, $FV(x)$ denotes the set of the free variables of x . In the rest of this paper, the sets of *closed terms*, *closed positive terms*, *closed negative terms* and *closed commands* are denoted respectively by \mathcal{T}^0 , \mathcal{T}_+^0 , \mathcal{T}_-^0 and \mathcal{C} . Moreover the set of values is denoted by \mathbb{V} .

2.2. Reduction

We now present the operational semantics for the syntax we just defined. The set of commands is equipped with the following one-step reduction relations \rightarrow_μ and \rightarrow_β :

$$\begin{array}{llll}
(+)& \langle \mu\alpha.c \mid t_- \rangle & \rightarrow_\mu & c[t_-/\alpha] \\
(-)& \langle V^+ \mid \mu x.c \rangle & \rightarrow_\mu & c[V^+/x] \\
(\uparrow)& \langle \{V\} \mid \mu\{\kappa\}.c \rangle & \rightarrow_\beta & c[V/\kappa] \\
(2\gamma)& \langle (V, V') \mid \mu(\kappa, \kappa').c \rangle & \rightarrow_\beta & c[V/\kappa, V'/\kappa']
\end{array}$$

(\rightarrow_β is defined only if the polarities of the κ 's and the V 's match)

We pose $\rightarrow_0 = \rightarrow_\beta \cup \rightarrow_\mu$.

Remarks 3.

1. The grammar defining the term syntax does not prevent ill-formed commands to appear. Indeed, consider the command $\langle \mu\alpha.c \mid \{V\} \rangle$. $\{V\}$ is a positive term whereas α is a negative variable. Hence, this command won't reduce. The possibility of this kind of ill-formed commands and terms will be removed by typing.
2. Even if the term syntax does not allow directly to form the pair (t, u) or the one-tuple $\{t\}$ when t and u are not values, it is possible to *define* these constructions as follows:

$$\begin{aligned}
(t, u) &= \mu\alpha.\langle t \mid \mu\kappa.\langle u \mid \mu\kappa'.\langle (\kappa, \kappa') \mid \alpha \rangle \rangle \rangle \\
\{t\} &= \mu\alpha.\langle t \mid \mu\kappa.\langle \{\kappa\} \mid \alpha \rangle \rangle
\end{aligned}$$

where the polarities of κ and κ' respectively match those of t and u . In the case of the pair, this definition reflects an arbitrary choice in the order of evaluation of t and u (here from left to right).

Definition 4 (Evaluation relation). Similarly to [22], we consider an *evaluation relation* to be a binary relation \rightarrow between commands such that $\rightarrow_0 \subseteq \rightarrow$.

In the rest paper, \rightarrow always denotes such an evaluation relation.

Remark 5. The fact that \rightarrow is not fixed will allow us to consider reduction rules when we extend the term syntax with new instructions, without losing the properties and theorems already proved.

Definition 6. Suppose \rightarrow is a binary relation between commands. If c is normalizing for $\rightarrow \cup \rightarrow_\mu$, then we define $\mathbf{Time}^\rightarrow(c)$ as the number of \rightarrow steps needed by c to normalize using \rightarrow and \rightarrow_μ . Otherwise, $\mathbf{Time}^\rightarrow(c)$ is undefined.

2.3. Kinds and type constructors

Here is exposed the language of **MAL** ω types. We define two syntactic categories: *kinds* and *type constructors* (or simply *constructors*).

$$\begin{array}{ll}
\textbf{Kinds} & \sigma, \tau ::= \iota \mid o^+ \mid o^- \mid \sigma \rightarrow \tau \\
\\
\textbf{Constructors} & A, B, T, U ::= x^\tau \mid x^{\tau^\perp} \mid \lambda x^\tau. T \mid TU \\
& \quad \mid \mathbf{0} \mid \mathbf{s} \mid \text{rec}_\tau \mid \text{rec}_\tau^\perp \\
& \quad \mid A \otimes B \mid A \wp B \mid \exists x^\tau. A \mid \forall x^\tau. A \\
& \quad \mid \downarrow A \mid \uparrow A
\end{array}$$

Kinds are a simple type system for constructors: ι is the kind representing *individuals*, $\sigma \rightarrow \tau$ is the kind of *functions* from σ to τ , o^+ is the kind of *positive formulas* and o^- the kind of *negative formulas*. We denote by \mathbf{n} the constructor $\mathbf{s}^n \mathbf{0}$.

Definition 7 (Involutive negation). The operation $(.)^\perp$ (called *negation*) is only defined on atomic constructors (variables and recursor rec_τ). It is extended as an involutive operation on all kinds as follows:

$$\begin{array}{ll}
o^{+\perp} &= o^- & o^{-\perp} &= o^+ \\
\iota^\perp &= \iota & (\sigma \rightarrow \tau)^\perp &= \sigma \rightarrow \tau^\perp
\end{array}$$

But also on all constructors:

$$\begin{array}{ll}
(x^\tau)^\perp &= x^{\tau^\perp} & (x^{\tau^\perp})^\perp &= x^\tau \\
\mathbf{0}^\perp &= \mathbf{0} & \mathbf{s}^\perp &= \mathbf{s} \\
(\lambda x^\tau. T)^\perp &= \lambda x^{\tau^\perp}. (T)^\perp & (TU)^\perp &= T^\perp U \\
(\text{rec}_\tau)^\perp &= \text{rec}_\tau^\perp & (\text{rec}_\tau^\perp)^\perp &= \text{rec}_\tau \\
(A \otimes B)^\perp &= A^\perp \wp B^\perp & (A \wp B)^\perp &= A^\perp \otimes B^\perp \\
(\forall x^\tau. A)^\perp &= \exists x^{\tau^\perp}. A^\perp & (\exists x^\tau. A)^\perp &= \forall x^{\tau^\perp}. A^\perp \\
(\uparrow A)^\perp &= \downarrow A^\perp & (\downarrow A)^\perp &= \uparrow A^\perp
\end{array}$$

The operation $(.)^\perp$ is involutive: for any constructor T , we have $T^{\perp\perp} = T$.

The rules of figure 1 define what it means for a constructor T to be of kind τ (and we note it $T : \tau$). When we write $T : o$ it means that $T : o^+$ or $T : o^-$. We say that a constructor T is *well-formed* if there exists some kind σ such that $T : \sigma$ holds.

Property 8. *If $T : \sigma$ then $T^\perp : \sigma^\perp$.*

Finally, we define a relation of *convertibility* between constructors, noted $T \cong T'$, whose inductive definition is given in Figure 2. Notice that if A and B are formulas and $A \cong B$ then A and B have the same polarity. The presence of the dual recursor rec_τ^\perp and its associated conversion rules are necessary to obtain the following property.

$$\begin{array}{c}
\frac{}{\overline{x^\tau : \tau}} \quad \frac{}{\overline{(x^\tau)^\perp : \tau^\perp}} \\
\frac{T : \tau}{\lambda x^\sigma . T : \sigma \rightarrow \tau} \quad \frac{T : \sigma \rightarrow \tau \quad U : \sigma}{TU : \tau} \\
\frac{}{\overline{\mathbf{0} : \iota}} \quad \frac{}{\overline{\mathbf{s} : \iota \rightarrow \iota}} \\
\frac{}{\overline{rec_\tau : \tau \rightarrow (\iota \rightarrow \tau \rightarrow \tau) \rightarrow \iota \rightarrow \tau}} \\
\frac{}{\overline{rec_\tau^\perp : \tau \rightarrow (\iota \rightarrow \tau \rightarrow \tau) \rightarrow \iota \rightarrow \tau}} \\
\frac{A : o \quad B : o}{A \otimes B : o^+} \quad \frac{A : o \quad B : o}{A \wp B : o^-} \quad \frac{A : o}{\downarrow A : o^+} \quad \frac{A : o}{\uparrow A : o^-} \\
\frac{A : o^* \quad * \in \{+, -\}}{\forall x^\tau . A : o^*} \quad \frac{A : o^* \quad * \in \{+, -\}}{\exists x^\tau . A : o^*}
\end{array}$$

Figure 1: Typing rules for constructors

Property 9. *If T and U are constructors such that $T \cong U$, then $T^\perp \cong U^\perp$.*

Remark 10. On the formulas constructor (of kind o^* for $* \in \{+, -\}$), the negation $(.)^\perp$ is the usual involutive negation of linear logic. However, on closed individuals, it is simply the identity (modulo \cong). For example $((\lambda x^\iota . \mathbf{s} x^\iota) \mathbf{0})^\perp \cong \mathbf{s} \mathbf{0}$.

Let us give a few examples of useful constructors we can define.

- The negation operator on positive formulas can be defined as the constructor $\lambda x^{o^+} . (x^{o^+})^\perp : o^+ \rightarrow o^-$. Notice that the dual variable $(x^{o^+})^\perp$ is bound by the lambda binder λx^{o^+} .
- If we define $U = \lambda z^{o^+} . rec_o z^{o^+} (\lambda x^{o^+} . \lambda y^\iota . \downarrow (x^{o^+})^\perp)$, we have

$$U A (\mathbf{s} n) \cong \downarrow (U A n)^\perp$$

For example,

$$\begin{aligned}
U A \mathbf{5} &\cong \downarrow \uparrow \downarrow \uparrow A^\perp \\
U A \mathbf{4} &\cong \downarrow \uparrow \downarrow \uparrow A
\end{aligned}$$

$$\begin{array}{c}
\frac{}{(\lambda x^\tau.T)(U) \cong T\{x^\tau := U\}} \qquad \frac{x^\tau \notin FV(T)}{\lambda x^\tau.Tx \cong T} \\
\\
\frac{}{rec_\tau TU 0 \cong T} \qquad \frac{}{rec_\tau TU (sn) \cong U n (rec_\tau TU n)} \\
\\
\frac{}{rec_\tau^\perp TU 0 \cong T^\perp} \qquad \frac{}{rec_\tau^\perp TU (sn) \cong U^\perp n (rec_\tau TU n)} \\
\\
\frac{}{T \cong T} \quad \frac{T \cong T'}{T' \cong T} \quad \frac{T \cong T' \quad T' \cong T''}{T \cong T''} \\
\\
\frac{T \cong T'}{\lambda x^\tau.T \cong \lambda x^\tau.T'} \quad \frac{T \cong T' \quad U \cong U'}{TU \cong T'U'} \\
\\
\frac{A \cong A' \quad B \cong B'}{A \otimes B \cong A' \otimes B'} \quad \frac{A \cong A' \quad B \cong B'}{A \wp B \cong A' \wp B'} \\
\\
\frac{A \cong A'}{\exists x^\tau A \cong \exists x^\tau A'} \quad \frac{A \cong A'}{\forall x^\tau A \cong \forall x^\tau A'}
\end{array}$$

Figure 2: Convertibility relation between constructors

In the rest of the paper, we designate by the letters N, M, \dots negative formulas and by the letters P, Q, \dots positive formulas. We designate by the letters A, B, \dots formulas of any polarity (positive or negative). Negative formulas N are intended to type negative terms (lazy terms), whereas positive formulas P will be used to type positive terms (eager terms). The modality \uparrow is used to turn a positive term into a negative one, that is transforms an eager term into a lazy one. \downarrow does just the contrary, that is turning a negative term into a positive one.

Remark 11. In contrast with [23], \forall and \exists do not change the polarity of the formula. This choice is made to keep realizers of existential and universal statements simpler, especially when we will define forcing in section 5.

2.4. Type system

The type system **MAL** ω relates terms of \mathbf{L}_{foc} with **MAL** ω formulas. *Typing contexts* (denoted by the symbols: $\Gamma, \Gamma', \Delta, \dots$) are finite sets containing

elements of the form $x : N$ or $\alpha : P$. *Typing judgments* are of the form:

$$\vdash t_+ : P \mid \Gamma \quad \text{or} \quad \vdash t_- : N \mid \Gamma \quad \text{or} \quad c : (\vdash \Gamma)$$

The rules of **MAL** ω are described in Figure 3. Notice that in **MAL** ω , only *affine* terms are typable. That means that every bound variable κ appears at most once in the command under the binder.

Remarks 12.

1. We have chosen to have weakening and conversion rules expressed only on commands. The reason is that weakening and conversion rules for terms are derivable from these two rules, using the cut and activation rules. For example, here is the derived rule (\cong) on terms (the case of weakening is similar):

$$\frac{\frac{\vdash t : A \mid \Gamma \quad \frac{\vdash \kappa : A^\perp \mid \kappa : A}{(Ax)} (Cut)}{\langle t \mid \kappa \rangle : (\vdash A, \Gamma)} \quad A \cong B}{\frac{\langle t \mid \kappa \rangle : (\vdash B, \Gamma)}{\vdash \mu\kappa.\langle t \mid \kappa \rangle : B \mid \Gamma} (\mu)} (\cong)$$

2. We can only form pairs of values (V, V') . This is reflected in the type system by the (\otimes) rule that introduces only such pairs. However, we can obtain a derived rule for the following definition of (t, t') already presented in Subsection 2.2:

$$(t, t') = \mu\alpha.\langle t \mid \mu x.\langle u \mid \mu y.\langle \alpha \mid (x, y) \rangle \rangle \rangle$$

We just give the partial derivation corresponding to the derived rule, leaving the easy part to the reader.

$$\frac{\frac{\vdash \mathbf{u} : \mathbf{B} \mid \Delta \quad \frac{\frac{\vdash \langle (x, y) \mid \alpha \rangle : (\vdash x : A^\perp, y : B^\perp, \alpha : A \otimes B)}{\mu y.\langle (x, y) \mid \alpha \rangle : B^\perp \mid x : A^\perp, \alpha : A \otimes B} (\mu)}{\langle u \mid \mu y.\langle (x, y) \mid \alpha \rangle \rangle : (\vdash x : A^\perp, \alpha : A \otimes B, \Delta)} (Cut)}{\vdash \mu x.\langle u \mid \mu y.\langle (x, y) \mid \alpha \rangle \rangle : A^\perp \mid \Delta, \alpha : A \otimes B} (\mu)}{\frac{\langle t \mid \mu x.\langle u \mid \mu y.\langle (x, y) \mid \alpha \rangle \rangle \rangle : (\vdash \alpha : A \otimes B, \Gamma, \Delta)}{\vdash (t, \mathbf{u}) : \mathbf{A} \otimes \mathbf{B} \mid \Gamma, \Delta} (\mu)} (Cut)$$

The same remark also holds for the construction $\{t\}$ defined in Subsection 2.2.

$$\begin{array}{c}
\frac{}{\vdash x : P \mid x : P^\perp} (Ax_+) \quad \frac{}{\vdash \alpha : N \mid \alpha : N^\perp} (Ax_-) \\
\\
\frac{c : (\vdash \alpha : P, \Gamma)}{\vdash \mu\alpha.c : P \mid \Gamma} (\mu_+) \quad \frac{c : (\vdash x : N, \Gamma)}{\vdash \mu x.c : N \mid \Gamma} (\mu_-) \\
\\
\frac{\vdash t : A \mid \Gamma \quad \vdash u : A^\perp \mid \Delta}{\langle t \mid u \rangle : (\vdash \Gamma, \Delta)} (Cut) \\
\\
\frac{\vdash V : A \mid \Gamma \quad \vdash V' : B \mid \Delta}{\vdash (V, V') : A \otimes B \mid \Gamma, \Delta} (\otimes) \quad \frac{c : (\vdash \kappa : A, \kappa' : B, \Gamma)}{\vdash \mu(\kappa, \kappa').c : A \wp B \mid \Gamma} (\wp) \\
\\
\frac{\vdash V : A \mid \Gamma}{\vdash \{V\} : \downarrow A \mid \Gamma} (\downarrow) \quad \frac{c : (\vdash \kappa : A, \Gamma)}{\vdash \mu\{\kappa\}.c : \uparrow A \mid \Gamma} (\uparrow) \\
\\
\frac{T : \tau \quad \vdash t : A[T/x^\tau] \mid \Gamma}{\vdash t : \exists x^\tau.A \mid \Gamma} (\exists) \quad \frac{\vdash V : A \mid \Gamma \quad x \text{ does not appear in } \Gamma}{\vdash V : \forall x^\tau.A \mid \Gamma} (\forall) \\
\\
\frac{c : (\vdash \kappa : A, \Gamma) \quad A \cong B}{c : (\vdash \kappa : B, \Gamma)} (\cong) \quad \frac{c : (\vdash \Gamma)}{c : (\vdash x : A, \Gamma)} (W)
\end{array}$$

Figure 3: Typing rules of **MAL** ω

2.5. Generalities on Call-by-value and Call-by-name

In this focalising version of **MAL** ω , it is possible to encode both call-by-name and call-by-value affine λ -calculi, as explained in [23]. Moreover, each β -reduction step in these calculi induces a constant number of reduction steps in the corresponding encoding.

Call-by-name λ -calculus — We consider the call-by-name affine λ -calculus, that is such that in every term $\lambda x.t$, x appears at most once in t . We exhibit an encoding of this calculus by giving a typed translation of the affine λ -calculus in the negative fragment of **MAL** ω . The implication \multimap is defined as follows:

$$N \multimap M \equiv N^\perp \wp M$$

Terms t and stacks π of the Krivine abstract machine [13] are encoded respectively using negative terms $|t\rangle$ and positive terms $\langle\pi|$, as follows:

$$\begin{aligned}\lambda\alpha.t_- &\equiv \mu(\alpha, x).\langle t_- | x \rangle \\ u.\pi &\equiv (u, \pi) \\ (t_-)u_- &\equiv \mu x.\langle t_- | u_-.x \rangle\end{aligned}$$

We can check that these definitions indeed implement Krivine Machine weak call-by-name reduction, as shown by the following reduction:

$$\begin{aligned}\langle (\lambda\alpha.t)u | \pi \rangle &= \langle \mu x.\langle \mu(\alpha, y).\langle t | y \rangle | u.x \rangle | \pi \rangle \\ &\rightarrow_0 \langle \mu(\alpha, y).\langle t | y \rangle | (u, \pi) \rangle \\ &\rightarrow_0 \langle t[u/\alpha] | \pi \rangle\end{aligned}$$

We see that each β -reduction step in the weak call-by-name λ -calculus corresponds exactly to two \rightarrow_0 reduction steps in this encoding.

Call-by-value λ -calculus — The call-by-value affine λ -calculus is obtained by taking a positive encoding of the implication:

$$P \multimap Q \equiv \downarrow (P^\perp \wp Q)$$

We define terms and environments using respectively positive and negative terms:

$$\begin{aligned}\lambda x.t_+ &\equiv \{\mu(x, \alpha).\langle t_+ | \alpha \rangle\} \\ (t_+)u_+ &\equiv \mu\alpha.\langle t_+ | u_+.\alpha \rangle \\ u.e &\equiv \mu\{\alpha\}.\langle \alpha | (u, e) \rangle\end{aligned}$$

It can be checked that we retrieve Curien-Herbelin $\bar{\lambda}\mu\tilde{\mu}_v$ calculus [5]. Here is the typical example of a reduction in the encoded calculus:

$$\begin{aligned}\langle (\lambda x.t)u | E \rangle &= \langle \mu\alpha.\langle \{\mu(x, \alpha').\langle t | \alpha' \rangle\} | u.\alpha \rangle | E \rangle \\ &\rightarrow_0 \langle \{\mu(x, \alpha').\langle t | \alpha' \rangle\} | \mu\{\alpha\}.\langle \alpha | (u, E) \rangle \rangle \\ &\rightarrow_0 \langle \mu(x, \alpha').\langle t | \alpha' \rangle | (u, E) \rangle \\ &\rightarrow_0 \langle u | \mu\kappa_1.\langle E | \mu\kappa_2.\langle (\kappa_1, \kappa_2) | \mu(x, \kappa).\langle t | \kappa \rangle \rangle \rangle \\ &\rightarrow_0^* \langle V | \mu\kappa_1.\langle E | \mu\kappa_2.\langle (\kappa_1, \kappa_2) | \mu(x, \kappa).\langle t | \kappa \rangle \rangle \rangle \\ &\rightarrow_0 \langle E | \mu\kappa_2.\langle (V, \kappa_2) | \mu(x, \kappa).\langle t | \kappa \rangle \rangle \rangle \\ &\rightarrow_0 \langle (V, E) | \mu(x, \kappa).\langle t | \kappa \rangle \rangle \\ &\rightarrow_0 \langle t[V/x] | E \rangle\end{aligned}$$

Here again, it is clear that to each step in the Curien-Herbelin $\bar{\lambda}\mu\tilde{\mu}_v$ calculus corresponds a constant number of steps in \mathbf{L}_{foc} .

3. Quantitative Krivine's realizability

In this section, we define the quantitative classical realizability for $\mathbf{MAL}\omega$. This construction is a direct extension of Munch's focalised version of Krivine's classical realizability [23] and integrates the quantitative aspects of [18]. In (non-quantitative) Krivine's classical realizability, formulas are interpreted as sets of terms closed by a notion of biorthogonality, and a realizability relation $t \Vdash A$ between terms and formulas is defined. In that setting, $t \Vdash A$ intuitively means “*t is a term whose computational behavior follows the specification A*”. In our work, we interpret formulas A as sets of pairs (t, p) where t is a term and p is an abstract quantity. The realizability relation becomes $(t, p) \Vdash A$, with the informal meaning “*t is a term whose computational behavior follows the specification A and uses during its execution a quantity of resources bounded by p*”. The presence of this abstract quantity allows us to build a quantitative extension of the well-known technique of reducibility candidates, which we call *quantitative reducibility candidates*. We use these to prove bounded-time termination results on typable terms.

3.1. Quantitative monoid

We introduce the notion of *quantitative monoid*, whose elements can be thought as resource quantities (like time, space or energy). Quantitative monoids are a generalization and a simplification of Dal Lago and Hofmann's resource monoids [18].

Definition 13. A *quantitative monoid* is a structure $(\mathcal{M}, +, \mathbf{0}, \leq, \|\cdot\|)$ where:

- $(\mathcal{M}, +, \mathbf{0}, \leq)$ is a preordered commutative monoid.
- $\|\cdot\| : \mathcal{M} \rightarrow \mathbb{N}$ is a function such that:
 - for every $p, q \in \mathcal{M}$, we have $\|p\| + \|q\| \leq \|p + q\|$.
 - Moreover, $\|\cdot\|$ is compatible with \leq , that is if $p \leq q$ then $\|p\| \leq \|q\|$.

If moreover, there is an element $\mathbf{1} \in \mathcal{M}$ such that $1 \leq \|\mathbf{1}\|$, then we say that $(\mathcal{M}, +, \mathbf{0}, \mathbf{1}, \leq, \|\cdot\|)$ is a *quantitative monoid with unit*.

From now on, we will often denote a quantitative monoid by its carrier \mathcal{M} and we use lower-case consonnes letters p, q, m, v, \dots to denote its elements. Moreover, if $n \in \mathbb{N}$ then we use the notation $n.p$ for $\underbrace{p + p + \dots + p}_{n \text{ times}}$.

Remarks 14.

1. If we think of elements $p, q \in \mathcal{M}$ as abstract quantities bounding respectively the resource consumption of programs t and u , then doing the operation $p + q$ can be seen as way to calculate a bound for the resource consumption of the process $\langle t \mid u \rangle$ resulting of the interaction of these two programs.
2. The intuition behind the anti-triangular inequality, $\|p\| + \|q\| \leq \|p + q\|$ is that the amount of resources potentially used by the interaction of two programs is more than the sum of the quantities of resources used by the two programs alone.
3. One corollary of the anti-triangular inequality is that $\|\mathbf{0}\| = 0$. Indeed,

$$2 \times \|\mathbf{0}\| = \|\mathbf{0}\| + \|\mathbf{0}\| \leq \|\mathbf{0}\|$$

Example 15. The structure $(\mathbb{N}, +, 0, 1, \leq, x \mapsto x)$ where $+$ is the usual addition on integers and \leq is the usual order on \mathbb{N} , is a quantitative monoid with unit.

Remark 16. In any quantitative monoid with unit \mathcal{M} we have elements of arbitrary big measure, that is for every $n \in \mathbb{N}$,

$$n \leq \|n.\mathbf{1}\|$$

It has to be noted that every resource monoid in the sense of [18] defines a quantitative monoid with unit, by choosing $\|p\| = \mathcal{D}(\mathbf{0}, p)$ where $\mathcal{D}(\cdot, \cdot)$ is the distance of the resource monoid.

3.2. Quantitative pole and orthogonality

Krivine's classical realizability is a framework parametrized by a set \perp of commands called the *pole*. This set can be seen as the set of *correct* processes, that is the notion of correctness we want to study. For example, \perp can be the set of normalizing commands. This set then induces a notion of orthogonality, which is used to define an interpretation of the type system.

Similarly, our model of **MAL** ω will be parametrized by a structure called *quantitative pole* that we defined now.

Definition 17. Let \mathcal{M} be a quantitative monoid. We define the notions of *weighted terms* and *weighted commands* as follows:

1. A *weighted term* is a pair $(t, p) \in \mathcal{T}^0 \times \mathcal{M}$.
2. A *weighted command* is a pair $(c, p) \in \mathcal{C} \times \mathcal{M}$.

Informally, a weighted command (c, p) carries a quantitative information p that is a bound on the amount of resources used by c during its execution. We also define the notion of *quantitative pole*, which will be the main parameter of our model.

Definition 18 (Quantitative pole). A *quantitative pole* is a pair (\mathcal{M}, \perp) where:

- \mathcal{M} is a quantitative monoid.
- $\perp \subseteq \mathcal{C} \times \mathcal{M}$ is a set of weighted commands.

When it is clear from the context, we will often refer to a quantitative pole using its set \perp .

As we will see, not all quantitative poles yield sound interpretations of \mathbf{MAL}_ω . We define a subclass of quantitative poles, the *saturated quantitative poles*.

Definition 19 (Saturated pole). A *saturated quantitative pole* is a structure $(\mathcal{M}, \perp, p_\beta)$ given by:

- A quantitative pole (\mathcal{M}, \perp) .
- An element p_β of \mathcal{M} .
- Moreover \perp satisfies the following properties:
 - (\rightarrow_β -saturation)** If $c \rightarrow_\beta c'$ and $(c', p) \in \perp$ then $(c, p + p_\beta) \in \perp$
 - (\rightarrow_μ -saturation)** If $c \rightarrow_\mu c'$ then $(c', p) \in \perp \iff (c, p) \in \perp$
 - (\leq -saturation)** If $p \leq p'$ and $(c, p) \in \perp$ then $(c, p') \in \perp$.

Remark 20. The element p_β corresponds informally to the *cost* of a single β -reduction step, as witnessed by the \rightarrow_β -saturation property. The \rightarrow_μ steps, however, are not considered as a resource cost, since they are mainly administrative reductions.

Example 21. • Suppose P is a *non-quantitative saturated pole*, i.e a set of commands which is closed under anti-evaluation (i.e: $c \rightarrow_0 c' \wedge c' \in P \Rightarrow c \in P$). If \mathcal{M} is a quantitative monoid and $p_\beta \in \mathcal{M}$, then $(\mathcal{M}, P \times \mathcal{M}, p_\beta)$ is a saturated quantitative pole.

- An important example is \perp_{Time} the set of bounded time terminating processes, namely $\perp_{Time} = \{ (c, p) \mid \mathbf{Time}(c) \leq \|p\| \}$. In particular all $(c, p) \in \perp_{Time}$ are such that c terminates. If \mathcal{M} has a unit $\mathbf{1}$, then \perp_{Time} provides a saturated quantitative pole by choosing $p_\beta = \mathbf{1}$. The \rightarrow_β -saturation property relies on the fact that if $c \rightarrow_\beta c'$ and $(c', p) \in \perp_{Time}$, then

$$\mathbf{Time}(c) = \mathbf{Time}(c') + 1 \leq \|p\| + 1 \leq \|p\| + \|\mathbf{1}\| \leq \|p + \mathbf{1}\|$$

Until the rest of this section, we assume a choice of a quantitative pole (\mathcal{M}, \perp) (which is not necessarily saturated). This quantitative pole \perp induces a notion of *orthogonality* between elements of $\mathcal{T}_+^0 \times \mathcal{M}$ and $\mathcal{T}_-^0 \times \mathcal{M}$.

Definition 22 (Orthogonality). We say that $(t_+, p) \in \mathcal{T}_+^0 \times \mathcal{M}$ and $(t_-, q) \in \mathcal{T}_-^0 \times \mathcal{M}$ are *orthogonal* and we note:

$$(t_+, p) \perp (t_-, q) \iff (\langle t_+ \mid t_- \rangle, p + q) \in \perp$$

This orthogonality relation is lifted as an operation on set of bounded terms. If $X \subseteq \mathcal{T}_+^0 \times \mathcal{M}$, then we define its orthogonal as

$$X^\perp \equiv \{ (t_-, q) \mid \forall (t_+, p) \in X, t_+ \perp t_- \}$$

In a similar way, if $X \subseteq \mathcal{T}_-^0 \times \mathcal{M}$, then

$$X^\perp \equiv \{ (t_+, p) \mid \forall (t_-, q) \in X, t_+ \perp t_- \}$$

Remark 23. Informally, the meaning of $(t_+, p) \perp (t_-, q)$ is that the interaction $\langle t_+ \mid t_- \rangle$ *behaves well* and uses an amount of resources *bounded* by $p + q$.

The operation $(\cdot)^\perp$ satisfies the usual properties of orthogonality.

Property 24. If X and Y are both subsets of $\mathcal{T}_+^0 \times \mathcal{M}$ (resp. subsets of $\mathcal{T}_-^0 \times \mathcal{M}$) then we have:

- $X \subseteq X^{\perp\perp}$
- $X \subseteq Y$ implies $Y^\perp \subseteq X^\perp$
- $X^{\perp\perp\perp} = X^\perp$

Property 25. If $(X_i)_{i \in I}$ is a family of subsets of $\mathcal{T}_+^0 \times \mathcal{M}$ (resp. $\mathcal{T}_-^0 \times \mathcal{M}$), then the following equalities hold:

1. $(\bigcup_{i \in I} X_i)^\perp = \bigcap_{i \in I} X_i^\perp$
2. $(\bigcap_{i \in I} X_i)^\perp = (\bigcup_{i \in I} X_i^\perp)^{\perp\perp}$

Finally, we define a few notations. Let $X \in \mathcal{P}(\mathcal{T}_+^0 \times \mathcal{M}) \cup \mathcal{P}(\mathcal{T}_-^0 \times \mathcal{M})$. Then:

$$\begin{aligned} X_{\mathbb{V}} &= X \cap (\mathbb{V} \times \mathcal{M}) \\ \overline{X} &= \{ (t, q) \mid \exists p \leq q \text{ such that } (t, p) \in X \} \end{aligned}$$

Remark 26. If $X \in \mathcal{P}(\mathcal{T}_-^0 \times \mathcal{M})$ then $X_{\mathbb{V}} = X$. Indeed, every negative term is also a value.

3.3. Interpretation of kinds

Before giving the actual interpretation of kinds and constructors, we define two operations on sets of bounded terms.

$$\begin{aligned} X \otimes Y &= \{ ((t, u), p + q) \mid (t, p) \in X \wedge (u, q) \in Y \} \\ \downarrow X &= \{ (\{t\}, p) \mid (t, p) \in X \} \end{aligned}$$

Suppose $\mathcal{D}^\oplus \in \mathcal{P}(\mathcal{T}_+^0 \cap \mathbb{V} \times \mathcal{M})$. We then define \mathcal{D}^\ominus as the set $\{ X^\perp \mid X \in \mathcal{D}^\oplus \}$ and we pose $\mathcal{D} = \mathcal{D}^\oplus \cup \mathcal{D}^\ominus$.

Definition 27 (Propositional domain). We say that \mathcal{D}^\oplus is a *positive propositional domain* if it satisfies the following properties:

- If $(X_i)_{i \in I}$ is a family of elements of \mathcal{D}^\oplus indexed by I , then $\bigcup_{i \in I} X_i \in \mathcal{D}^\oplus$ and $\bigcap_{i \in I} X_i \in \mathcal{D}^\oplus$.
- If $X, Y \in \mathcal{D}$, then $X \otimes Y \in \mathcal{D}^\oplus$.
- If $X \in \mathcal{D}$, then $\downarrow X \in \mathcal{D}^\oplus$.

Now suppose we have fixed a positive propositional domain \mathcal{D}^\oplus . We begin with the interpretation of kinds. If σ is a kind we define its *interpretation* $\|\sigma\|$:

$$\begin{aligned} \|\iota\| &= \mathbb{N} \\ \|o^+\| &= \mathcal{D}^\oplus \\ \|o^-\| &= \mathcal{D}^\ominus \\ \|\sigma \rightarrow \tau\| &= \|\tau\|^{\|\sigma\|} \end{aligned}$$

3.4. Interpretation of constructors

The orthogonality operation $(.)^\perp$ defined earlier on set of bounded terms, is extended inductively on elements of all kinds. That is, for T being an element of $\|\sigma\|$, we define $\perp(T, \sigma)$ as:

$$\begin{aligned} \perp(X, \iota) &= X \\ \perp(X, o^+) &= X^\perp \\ \perp(X^\perp, o^-) &= X \\ \perp(X, \sigma \rightarrow \tau) &= Y \in \|\sigma\| \mapsto \perp(X(Y), \tau) \end{aligned}$$

Notice that this definition makes sense only because \mathcal{D}^\oplus is a positive propositional domain. Hence we know that any element of $\|o^-\|$ is the orthogonal of an element of $\|o^+\|$.

Example 28. If $X \in \mathcal{D}^\oplus$, then $\perp(X, o^+)$ coincide with the orthogonal X^\perp of X . On the kind $o^+ \rightarrow o^+$, consider for example $X \mapsto X \in \|o^+ \rightarrow o^+\|$, we obtain $\perp(X \mapsto X, o^+ \rightarrow o^+) = X \mapsto X^\perp \in \|o^+ \rightarrow o^-\|$, that is the orthogonality operator.

This notion of extended orthogonality is consistent with the syntactic orthogonality on kinds, as witnessed by the following property.

Property 29. *If $T \in \|\sigma\|$, then we have $\perp(T, \sigma) \in \|\sigma^\perp\|$.*

PROOF. It is proved by induction on the kind σ , and is a consequence of the definition of σ^\perp and the fact that \mathcal{D}^\oplus is a propositional domain.

Given the positive propositional domain, a *valuation* is a partial function ρ assigning to a variable x^σ of kind σ an element $\rho(x^\sigma) \in \|\sigma\|$. We denote by $\rho[x^\sigma \leftarrow v]$ the valuation obtained from ρ by (re)binding the variable x^σ to the element $v \in \|\sigma\|$. We say that ρ *closes* a constructor T if $FV(T) \subseteq \text{dom}(\rho)$ and we note it $\rho \Vdash T$. By extension, we denote by $\rho \Vdash T_1, \dots, T_n$ if ρ closes each constructor T_i . A *total valuation* is a valuation whose domain is the set of all

higher-order variables. If ρ is a total valuation, then for every constructor T , $\rho \Vdash T$.

Given a well-typed constructor T and a valuation ρ such that $\rho \Vdash T$, we define the set $\|T\|_\rho$ by induction on T :

$$\begin{aligned}
\|x^\sigma\|_\rho &= \rho(x) \\
\|(x^\sigma)^\perp\|_\rho &= \perp(\rho(x), \sigma) \\
\|\lambda x^\sigma.T\|_\rho &= (v \in \|\sigma\| \mapsto \|T\|_{\rho[x \leftarrow v]}) \\
\|TU\|_\rho &= (\|T\|_\rho)\|U\|_\rho \\
\|\mathbf{0}\|_\rho &= \mathbf{0} \\
\|\mathbf{s}\|_\rho &= n \mapsto n + 1 \\
\|rec_\tau\|_\rho &= rec_{\|\tau\|} \\
\|rec_\tau^\perp\|_\rho &= \perp(rec_{\|\tau\|}, (\tau \rightarrow (\iota \rightarrow \tau \rightarrow \tau) \rightarrow \iota \rightarrow \tau))
\end{aligned}$$

Concerning constructors A which are formulas, that is of kind o^+ and o^- , the set $\|A\|_\rho$ is an element of $\mathcal{P}(\mathcal{T}_+^0 \times \mathcal{M}) \cup \mathcal{P}(\mathcal{T}_-^0 \times \mathcal{M})$. Moreover, it contains only values:

$$\begin{aligned}
\|\downarrow A\|_\rho &= \downarrow \|A\|_\rho \\
\|\uparrow A\|_\rho &= (\downarrow \|A^\perp\|_\rho)^\perp \\
\|A \otimes B\|_\rho &= \|A\|_\rho \otimes \|B\|_\rho \\
\|A \wp B\|_\rho &= (\|A^\perp\|_\rho \otimes \|B^\perp\|_\rho)^\perp \\
\|\exists x^\sigma.P\|_\rho &= \bigcup_{v \in \|\sigma\|} \|P\|_{\rho[x^\sigma \leftarrow v]} \\
\|\forall x^\sigma.N\|_\rho &= (\bigcup_{v \in \|\sigma\|} \|N^\perp\|_{\rho[x^\sigma \leftarrow v]})^\perp \\
\|\forall x^\sigma.P\|_\rho &= \bigcap_{v \in \|\sigma\|} \|P\|_{\rho[x^\sigma \leftarrow v]} \\
\|\exists x^\sigma.N\|_\rho &= (\bigcap_{v \in \|\sigma\|} \|N^\perp\|_{\rho[x^\sigma \leftarrow v]})^\perp
\end{aligned}$$

Finally, for a formula A , we define the set $|A|_\rho$ as

$$|A|_\rho = \|A\|_\rho^{\perp\perp}$$

Remarks 30.

1. For each well-typed constructor $T : \sigma$ and each valuation $\rho \Vdash T$, we have $\|T\|_\rho \in \|\sigma\|$. This rely on the fact that \mathcal{D}^\oplus is a positive propositional domain, and hence is closed under the required operations.
2. For the negative existential case, we notice that

$$\|\exists x^\tau.N\|_\rho = (\bigcup_{v \in \|\tau\|} \|N\|_{\rho[x^\tau \leftarrow v]})^{\perp\perp}$$

3. However, for the universal case, the interpretation of $\forall x^\tau.A$ is always

$$\|\forall x^\tau.A\|_\rho = \bigcap_{v \in \|\tau\|} \|A\|_{\rho[x^\tau \leftarrow v]}$$

even if A is negative. We gave different formulations for the negative and positive cases in order to show clearly that $\|\forall x^\tau.N\|_\rho = \|\exists x^\tau.N^\perp\|_\rho^\perp$, but this remark shows it is not mandatory.

If T is a closed well-typed constructor, $\|T\|_\rho$ and $|T|_\rho$ are independent of ρ . Hence we will often simply note them respectively $\|T\|$ and $|T|$.

Property 31. *The interpretation $\|\cdot\|$ enjoys the following properties.*

1. *If T and T' are two well-typed constructors such that $T \cong T'$, then for every valuation $\rho \Vdash T, T'$, we have $T \cong T'$ then $\|T\|_\rho = \|T'\|_\rho$.*
2. *For any constructor T of kind σ and $\rho \Vdash T$, we have $\|T^\perp\|_\rho = \perp(\|T\|_\rho, \sigma)$.*
3. *For any well-typed constructors $T : \tau$ and $S : \sigma$, any valuation ρ such that $\rho \Vdash S$ and $FV(T) \subseteq \text{dom}(\rho) \cup \{x^\sigma\}$, we have*

$$\|T\|_{\rho[x^\sigma \leftarrow \|S\|_\rho]} = \|T[S/x^\sigma]\|_\rho$$

PROOF. 1. This is immediate by induction first on the kind σ and on the judgment \cong .

2. This is proved by induction on the typing judgment of the constructor T .

3. This is proved by induction on the typing judgment of the constructor T .

Remark 32. Notice that neither the definition of the interpretation of well-typed constructors nor the proof of Property 31 need to suppose that \perp is saturated.

We say that (t, p) *realizes* a closed formula A and we note $(t, p) \Vdash^\rho A$ iff $(t, p) \in \|A\|_\rho$. If the choice of ρ is clear from the context we only note it $(t, p) \Vdash A$. We may sometimes use the notation $(t, p) \Vdash_\perp A$ to precise the quantitative pole we consider.

3.5. Properties of saturated quantitative poles

Until now, we have considered a quantitative pole which is not necessarily saturated. When the pole is saturated, we can derive many properties that will be crucial to prove that our model is sound with respect to **MAL** ω . In this subsection, we suppose that $(\mathcal{M}, \perp, p_\beta)$ is a saturated quantitative pole, and explore the properties satisfied by the orthogonality operation. The first property we prove expresses the fact that a set closed by biorthogonality is also \leq -saturated.

Property 33. *For every $X \in \mathcal{P}(\mathcal{T}_+^0 \times \mathcal{M}) \cup \mathcal{P}(\mathcal{T}_-^0 \times \mathcal{M})$, we have $\overline{X} \subseteq X^{\perp\perp}$.*

PROOF. Let $(t, p) \in X$ and $q \in \mathcal{M}$ such that $p \leq q$. If $(u, r) \in X^\perp$ then $(t, p) \perp (u, r)$. By \leq -saturation of \perp , we have $(t, q) \perp (u, r)$. Hence $(t, q) \in X^{\perp\perp}$.

The following lemma proves that we can safely remove or add double orthogonal operators in interpretations of constructors.

Lemma 34. Suppose $X, Y \in \mathcal{P}(\mathcal{T}_+^0 \times \mathcal{M}) \cup \mathcal{P}(\mathcal{T}_-^0 \times \mathcal{M})$ and $D \subseteq \mathcal{P}(\mathcal{T}_+^0 \times \mathcal{M}) \cup \mathcal{P}(\mathcal{T}_-^0 \times \mathcal{M})$ with $D \neq \emptyset$. Then we have the following equalities:

1. $(\downarrow X)^{\perp\perp} = (\downarrow X^{\perp\perp})^{\perp\perp}$
2. $(X \otimes Y)^{\perp\perp} = (X^{\perp\perp} \otimes Y^{\perp\perp})^{\perp\perp}$
3. $(\bigcap_{X \in D} X)^{\perp\perp} = \bigcap_{X \in D} X^{\perp\perp}$
4. $(\bigcup_{X \in D} X)^{\perp\perp} = (\bigcup_{X \in D} X^{\perp\perp})^{\perp\perp}$

PROOF. 1. Since $X \subseteq X^{\perp\perp}$, we immediately have $(\downarrow X)^{\perp\perp} \subseteq (\downarrow X^{\perp\perp})^{\perp\perp}$. Let's prove that $(\downarrow X^{\perp\perp})^{\perp\perp} \subseteq (\downarrow X)^{\perp\perp}$. It suffices to show that $(\downarrow X)^{\perp} \subseteq (\downarrow X^{\perp\perp})^{\perp}$. Let $(t, q) \in (\downarrow X)^{\perp}$ and $(u, p) \in X^{\perp\perp}$. We want to show that $(\langle t | \{u\} \rangle, p + q) \in \perp$. Two cases are possible:

- If u is a value, then by \rightarrow_μ -saturation of \perp , it suffices to show that $(\langle \mu\kappa. \langle t | \{\kappa\} \rangle | u \rangle, p + q) \in \perp$.
- If u is not a value then $\langle t | \{u\} \rangle \rightarrow_i \langle \mu\kappa. \langle t | \{\kappa\} \rangle | u \rangle$ and so by \rightarrow_i -saturation of \perp , it suffices to show that $(\langle \mu\kappa. \langle t | \{\kappa\} \rangle | u \rangle, p + q) \in \perp$.

In both cases it is a consequence of $(\mu\kappa. \langle t | \{\kappa\} \rangle, p) \in X^{\perp\perp} = X^\perp$, which is immediate because $(t, p) \in (\downarrow X)^\perp$ and by \rightarrow_μ -saturation of \perp .

2. Similarly we only have to prove that $(X \otimes Y)^\perp \subseteq (X^{\perp\perp} \otimes Y^{\perp\perp})^\perp$. Let $(t, q) \in (X \otimes Y)^\perp$ and $((u, u'), p + p') \in (X^{\perp\perp} \otimes Y^{\perp\perp})$. By the same argument of \rightarrow_μ and \rightarrow_i -saturation of \perp , it suffices to prove that $(\mu\kappa. \langle t | (\kappa, u') \rangle, q + p') \in X^\perp$. To do so let's take $(u'', p'') \in X$ and show $(\langle u'' | \mu\kappa. \langle t | (\kappa, u') \rangle \rangle, q + p' + p'') \in \perp$. Again, by \rightarrow_μ and \rightarrow_i saturation it suffices to show that $(\mu\kappa'. \langle t | (u'', \kappa') \rangle, q + p'') \in Y^\perp$. Again, take $(u''', p''') \in Y$. We have clearly $(\langle t | (u'', u''') \rangle, q + p'' + p''') \in \perp$ since $(t, q) \in (X \otimes Y)^\perp$. Hence our result.
3. This is immediate by Property 25.
4. This is immediate by Property 25.

Lemma 35. Let A a formula, t a term of the same polarity as A and which has exactly one free variable κ , $q \in \mathcal{M}$ and X a subset of $\mathcal{T}_+^0 \cap \mathbb{V} \times \mathcal{M}$ or of $\mathcal{T}_-^0 \times \mathcal{M}$. The following properties are equivalent:

- i) For each $(V, q) \in X$, $(c[V/\kappa], p + q) \in \perp$
- ii) For each $(V, q) \in X^{\perp\perp} \cap \mathbb{V}$, $(c[V/\kappa], p + q) \in \perp$

PROOF. • $(ii) \Rightarrow (i)$ This is immediate since $X \subseteq X^{\perp\perp} \cap \mathbb{V}$.

- $(i) \Rightarrow (ii)$ Suppose (i) . That means for every $(V, q) \in X$, by \rightarrow_μ saturation of \perp , $(\langle \mu\kappa. c | V \rangle, p + q) \in \perp$. Hence, $(\mu\kappa. c, p) \in X^\perp = X^{\perp\perp\perp}$, so for every $(V, q) \in X^{\perp\perp} \cap \mathbb{V}$, $(\langle \mu\kappa. c | V \rangle, p + q) \in \perp$ and by \rightarrow_μ closure of \perp and because V is a value, $(c[V/\kappa], p + q) \in \perp$.

Remark 36. This property, which is true for every choice of saturated quantitative pole, will be useful when we will extend our interpretation to stronger type systems (that can handle contraction). This lemma requires to work in a

calculus where substitution and interaction can be *exchanged* in the following sense:

$$(c[t/\kappa], p + q) \in \perp\!\!\!\perp \iff (\langle \mu\kappa.c \mid t \rangle, p + q) \in \perp\!\!\!\perp$$

In particular, it is impossible to have this property in the framework of the usual Krivine's realizability: only head contexts are considered while we need general contexts. This justifies the use of a completely symmetric calculus.

3.6. Adequacy

Before we can state and prove the soundness of our realizability interpretation with respect to **MAL** ω , we define what it means for a typing rule to be *adequate*. All the following notions are defined *with respect to some quantitative pole* $\perp\!\!\!\perp$.

By abuse, if $\Gamma = \kappa_1 : A_1, \dots, \kappa_n : A_n$ is a typing context and ρ is a valuation, we will write $\rho \Vdash \Gamma$ as a notation for $\rho \Vdash A_1, \dots, A_n$. We will also denote by $\Gamma[\rho]$ a pair (Γ, ρ) where $\rho \Vdash \Gamma$.

Definition 37 (Substitution). A *substitution* σ is a partial application from the set of term variables to the set $\mathbb{V} \times \mathcal{M}$, whose domain $\text{dom}(\sigma)$ is finite. We will note $[\kappa_1 \leftarrow (V_1, p_1), \dots, \kappa_n \leftarrow (V_n, p_n)]$ to denote the substitution σ where $\text{dom}(\sigma) = \{\kappa_1, \dots, \kappa_n\}$ and such that $\sigma(\kappa_i) = (V_i, p_i)$.

Suppose $\sigma = [\kappa_1 \leftarrow (V_1, p_1), \dots, \kappa_n \leftarrow (V_n, p_n)]$ is a substitution.

- If (c, q) is a bounded command, we note

$$(c, q)[\sigma] = (c[V_1/\kappa_1, \dots, V_n/\kappa_n], q + \sum_i p_i)$$

- If (u, q) is a bounded term then we note

$$(u, q)[\sigma] = (u[V_1/\kappa_1, \dots, V_n/\kappa_n], q + \sum_i p_i)$$

If σ is a substitution, we denote by $\sigma[\kappa \leftarrow (V, p)]$ the substitution obtained from σ by rebinding κ to (V, p) . If σ_1 is a substitution and $\sigma_2 = [\kappa_1 \leftarrow (V_1, p_1), \dots, \kappa_n \leftarrow (V_n, p_n)]$ is another substitution, we denote by

$$\sigma_1, \sigma_2 = (\dots (\sigma_1[\kappa_1 \leftarrow (V_1, p_1)]) \dots) [\kappa_n \leftarrow (V_n, p_n)]$$

Definition 38 (Adequate substitution). Let $\Gamma = \kappa_1 : A_1, \dots, \kappa_n : A_n$ be a context and $\rho \Vdash \Gamma$. We say that a substitution σ is *adequate* to $\Gamma[\rho]$ and we note $\sigma \Vdash \Gamma[\rho]$ iff

- $\text{dom}(\sigma) = \{\kappa_1, \dots, \kappa_n\}$
- $\forall i \in \llbracket 1, n \rrbracket, \sigma(\kappa_i) \in \|A_i^\perp\|_\rho$

In particular, if Γ is a typing context and σ a substitution adequate to Γ then for every negative (resp. positive) variable κ appearing in Γ , $\sigma(\kappa) \in \mathcal{T}_+^0 \cap \mathbb{V} \times \mathcal{M}$ (resp. $\mathcal{T}_-^0 \times \mathcal{M}$). Indeed, positive (resp. negative) variables of Γ are associated to negative (resp. positive) formulas.

Definition 39 (Adequate judgment). Suppose $\Gamma = \kappa_1 : A_1, \dots, \kappa_n : A_n$ is a context and $p \in \mathcal{M}$.

- A judgment of the form $c : (\vdash \Gamma)$ is said to be *p-adequate* iff for every total valuation ρ and for every adequate substitution $\sigma \Vdash \Gamma[\rho]$ we have $(c, p)[\sigma] \in \perp$.
- Similarly, a judgment of the form $\vdash t : B \mid \Gamma$ is said to be *p-adequate* iff for every total valuation ρ and for every adequate substitution $\sigma \Vdash \Gamma[\rho]$ we have $(t, p)[\sigma] \in |B|_\rho$. Moreover, if $t \in \mathbb{V}$ then $(t, p)[\sigma] \in \overline{\|B\|_\rho}$.

We have now everything we need to define what it means for our interpretation to be sound with respect to a given typing rule. A typing rule is given by a sequence of premises J_i (which are typing judgments), side-conditions (SC) on these judgments, and a conclusion K :

$$\frac{J_1 \quad J_2 \quad \dots \quad J_n \quad SC}{K} (rule)$$

Definition 40 (Adequate rule). Suppose R is a typing rule, with J_1, \dots, J_n being its premises judgments and K its conclusion. Suppose $f : \mathcal{M}^n \rightarrow \mathcal{M}$ is a n -ary function on the quantitative monoid. We say that:

R is *f-adequate* iff for every $p_1, \dots, p_n \in \mathcal{M}$, (for all $i \in \llbracket 1, n \rrbracket$, J_i is p_i -adequate) implies that K is $f(p_1, \dots, p_n)$ -adequate.

Remarks 41.

1. If a 0-ary rule (like the axiom rule) then the notion of *f-adequacy* makes sense only if f is an element of the quantitative monoid \mathcal{M} .
2. If a typing derivation π is built using only adequate rules, then its conclusion is also *p-adequate* for some $p \in \mathcal{M}$. That p is obtained by composing the functions associated to each rule accordingly to the derivation structure π .

We now prove an **adequacy theorem** that relates typing in **MAL** ω and quantitative realizability. We suppose having chosen a saturated quantitative pole $(\mathcal{M}, \perp, p_\beta)$ and a positive propositional domain \mathcal{D}^\oplus . We first associate to each **MAL** ω rule R a function $\mathbf{M}[R] : \mathcal{M} \rightarrow \mathcal{M}$. We will then show that each rule R of **MAL** ω is $\mathbf{M}[R]$ -adequate.

$$\begin{array}{ll} \mathbf{M}[Ax_+] = \mathbf{0} & \mathbf{M}[Ax_-] = \mathbf{0} \\ \mathbf{M}[\mu_-] = x \mapsto x & \mathbf{M}[\mu_+] = x \mapsto x \\ \mathbf{M}[\otimes] = (x, y) \mapsto x + y & \mathbf{M}[\wp] = x \mapsto x + p_\beta \\ \mathbf{M}[\downarrow] = x \mapsto x & \mathbf{M}[\uparrow] = x \mapsto x + p_\beta \\ \mathbf{M}[\exists] = x \mapsto x & \mathbf{M}[\forall] = x \mapsto x \\ \mathbf{M}[\cong] = x \mapsto x & \mathbf{M}[W] = x \mapsto x \\ \mathbf{M}[Cut] = (x, y) \mapsto x + y & \end{array}$$

Theorem 42. *Suppose that $(\mathcal{M}, \perp, p_\beta)$ is a saturated quantitative pole. Every rule R of $\mathbf{MAL}\omega$ is $\mathbf{M}[R]$ -adequate.*

PROOF. To prove this statement, we just look at each of the $\mathbf{MAL}\omega$ rules and check that they are adequate.

- (Ax_*) The proof is the same in the positive and negative cases. Let $(t, p) \in \|A^{\perp\perp}\| = \|A\|$, then $(\kappa, \mathbf{0})[\kappa \mapsto (t, p)] = (t, p) \in \|A\|_\rho \subseteq \overline{\|A\|_\rho}$. Hence the rule is $\mathbf{0}$ -adequate.
- (Cut) Suppose $\vdash t_+ : P \mid \Gamma$ is p -adequate and $\vdash t_- : P^\perp \mid \Delta$ is q -adequate. Let $\rho \Vdash \Gamma, \Delta$ and $\sigma \Vdash (\Gamma, \Delta)[\rho]$. We can split $\sigma = \sigma_1, \sigma_2$ such that $\sigma_1 \Vdash \Gamma[\rho]$ and $\sigma_2 \Vdash \Delta[\rho]$. We want to show that $(\langle t_+ \mid t_- \rangle, p + q)[\sigma] \in \perp$. For any $\rho' \Vdash P, \Gamma, \Delta$ whose restriction is ρ (such a ρ' exists), we have $(t_+, p)[\sigma_1] \in \|P\|_{\rho'}^{\perp\perp}$ and $(t_-, q)[\sigma_2] \in \|P^\perp\|_{\rho'} = \|P\|_{\rho'}^\perp$. Hence $(t_+, p)[\sigma_1] \perp (t_-, q)[\sigma_2]$, and so $(\langle t_+ \mid t_- \rangle, p + q)[\sigma] \in \perp$.
- (\otimes) We suppose $\vdash V : A \mid \Gamma$ is p -adequate and $\vdash V' : B \mid \Delta$ is q -adequate. Let $\rho \Vdash \Gamma, \Delta, A \otimes B$ and $\sigma \Vdash (\Gamma, \Delta)[\rho]$. Hence, σ can be split into $\sigma_1 \Vdash \Gamma[\rho]$ and $\sigma_2 \Vdash \Delta[\rho]$. Because $\rho \Vdash \Gamma, A$ and $\rho \Vdash \Delta, B$ we know by hypothesis that $(V, p)[\sigma_1] \in \overline{\|A\|_\rho}$ and $(V', q)[\sigma_2] \in \overline{\|B\|_\rho}$. Hence $((V, V'), (p + q'))[\sigma_1, \sigma_2] \in \overline{\|A\|_\rho \otimes \|B\|_\rho} = \overline{\|A \otimes B\|_\rho}$. Because $\sigma = \sigma_1, \sigma_2$, we can conclude that the (\otimes) rule is $\mathbf{M}[\otimes]$ -adequate.
- (\wp) Suppose $c : (\vdash \kappa : A, \kappa' : B, \Gamma)$ is p -adequate for some p . Let $\rho \Vdash \Gamma, A \wp B$ and $\sigma \Vdash \Gamma[\rho]$. We want to show that $(\mu(\kappa, \kappa').c, p + p_\beta)[\sigma] \in \|A \wp B\|_\rho$. Since $\|A \wp B\|_\rho = \|A^\perp \otimes B^\perp\|_\rho^\perp$ by Property 31, we take $(V, q) \in \|A^\perp\|_\rho$ and $(V', q') \in \|B^\perp\|_\rho$ and show that $(\langle \mu(\kappa, \kappa').c \mid (V, V') \rangle, p + p_\beta + q + q')[\sigma] \in \perp$. But it is easy to see that $\sigma, \kappa \mapsto (V, q), \kappa' \mapsto (V', q') \Vdash (\Gamma, \kappa : A, \kappa' : B)[\rho]$. Hence, because the premise is p -adequate we obtain $(c[V/\kappa, V'/\kappa'], p + q + q')[\sigma] \in \perp$. By \rightarrow_β -saturation of \perp , we finally obtain $(\langle \mu(\kappa, \kappa').c \mid (V, V') \rangle, p + p_\beta + q + q')[\sigma] \in \perp$.
- (\downarrow) We suppose $\vdash V : A \mid \Gamma$ is p -adequate. Let $\rho \Vdash \Gamma, \downarrow A$ and $\sigma \Vdash \Gamma[\rho]$. Because $\rho \Vdash \Gamma, A$ we know by hypothesis that $(V, p)[\sigma] \in \overline{\|A\|_\rho}$. Hence $(\{V\}, q)[\sigma] \in \downarrow \overline{\|A\|_\rho} = \overline{\downarrow \|A\|_\rho}$. We conclude that the (\downarrow) rule is $\mathbf{M}[\downarrow]$ -adequate.
- (\uparrow) Suppose $c : (\vdash \kappa : A, \Gamma)$ is p -adequate for some $p \in \mathcal{M}$. Let $\rho \Vdash \Gamma, \uparrow A$ and $\sigma \Vdash \Gamma[\rho]$. We want to show that $(\mu\{\kappa\}.c, p + p_\beta)[\sigma] \in \|\uparrow A\|_\rho$. Since $\|\uparrow A\|_\rho = \|\downarrow A^\perp\|_\rho^\perp$ by Property 31, we take $(V, q) \in \|A^\perp\|_\rho$ and show that $(\langle \mu\{\kappa\}.c \mid \{V\} \rangle, p + p_\beta + q)[\sigma] \in \perp$. But it is easy to see that $\sigma, \kappa \mapsto (V, q) \Vdash (\Gamma, \kappa : A)[\rho]$. Hence, because the premise is p -adequate we obtain $(c[V/\kappa], p + q)[\sigma] \in \perp$. By \rightarrow_β -saturation of \perp , we finally obtain $(\langle \mu\{\kappa\}.c \mid \{V\} \rangle, p + p_\beta + q)[\sigma] \in \perp$.

- (μ_+) Suppose $c : (\vdash \alpha : P, \Gamma)$ is p -adequate for some $p \in \mathcal{M}$. Let $\rho \Vdash P, \Gamma$ and $\sigma \Vdash \Gamma[\rho]$. We want to show that $(\mu\alpha.c, p)[\sigma] \in |P|_\rho = \|P\|_\rho^{\perp\perp}$. So we take $(u, q) \in \|P\|_\rho^\perp$ and want to conclude that $(\langle \mu\alpha.c \mid u \rangle, p + q)[\sigma] \in \perp$. But $\|P\|_\rho^\perp = \|P^\perp\|_\rho$ by Property 31. Hence, $\sigma, \alpha \mapsto (u, q) \Vdash (\alpha : P, \Gamma)[\rho]$. Since the premise of the rule is p -adequate we conclude that $(c[u/\alpha], p + q)[\sigma] \in \perp$. But $\langle \mu\alpha.c \mid u \rangle \rightarrow_\mu c[u/\alpha]$ because α and u are negative. Hence, by \rightarrow_μ -saturation of \perp we obtain $(\langle \mu\alpha.c \mid u \rangle, p + q)[\sigma] \in \perp$.
- (μ_-) Suppose $c : (\vdash x : N, \Gamma)$ is p -adequate for some $p \in \mathcal{M}$. Let $\rho \Vdash N, \Gamma$ and $\sigma \Vdash \Gamma[\rho]$. We want to show that $(\mu x.c, p)[\sigma] \in \|N\|_\rho = \|N^\perp\|_\rho^\perp$. Let $(u, q) \in \|N^\perp\|_\rho$. It is sufficient to show that $(\langle \mu x.c \mid u \rangle, p + q)[\sigma] \in \perp$. But it is immediate that $\sigma, x \mapsto (u, q) \Vdash (x : N, \Gamma)[\rho]$. Hence, because the premise is p -adequate, we obtain $(c[u/x], p + q)[\sigma] \in \perp$. Since u is a value (because $(u, q) \in \|N^\perp\|_\rho$), by \rightarrow_μ saturation we obtain $(\langle \mu x.c \mid u \rangle, p + q)[\sigma] \in \perp$.
- (W) Suppose $c : (\vdash \Gamma)$ is p -adequate. Let $\rho \Vdash \Gamma, A$ and $\sigma \Vdash (\Gamma, \kappa : A)[\rho]$. Then $\sigma = \sigma', \kappa \mapsto (u, q)$ with $\sigma' \Vdash \Gamma[\rho]$. But $\rho \Vdash \Gamma$ so we conclude that $(c, p)[\sigma'] \in \perp$. By \leq -saturation of \perp we obtain immediately $(c, p)[\sigma] \in \perp$.
- (\forall^τ) Suppose $\vdash V : A \mid \Gamma$ is p -adequate and x^τ does not appear free in Γ . We want to show that $\vdash V : \forall x^\tau. A \mid \Gamma$ is p -adequate. Let $\rho \Vdash \forall x^\tau. A, \Gamma$ and $\sigma \Vdash \Gamma[\rho]$. By Remarks 30 and because V is a value, whatever the polarity of A is, we have to show that $(V, p)[\sigma] \in \bigcap_{v \in \|\tau\|} \|A\|_{\rho[x^\tau \leftarrow v]}$. So let $v \in \|\tau\|$ and we pose $\rho' = \rho[x^\tau \leftarrow v]$. We have $\rho' \Vdash (A, \Gamma)$. Moreover, $\sigma \Vdash \Gamma[\rho']$ because $\sigma \Vdash \Gamma[\rho]$ and x^τ does not appear free in Γ . Hence by hypothesis, $(V, p)[\sigma] \in \|A\|_{\rho[x^\tau \leftarrow v]}$, which permits to conclude.
- (\exists^τ) Let's first handle the case of values. Suppose $\vdash V : A[T/x^\tau] \mid \Gamma$ is p -adequate for some $T : \tau$. We want to show that $\vdash V : \exists x^\tau. A \mid \Gamma$ is p -adequate. Let ρ be a total valuation and $\sigma \Vdash \Gamma[\rho]$. We can suppose that x^τ does not appear in Γ (if it does, then we can rename it in $\exists x^\tau. A$). Because of p -adequacy of the premise, we have $(V, p)[\sigma] \in \|A[T/x^\tau]\|_\rho$. But by Property 31, we have $\|A[T/x^\tau]\|_\rho = \|A\|_{\rho[x^\tau \leftarrow \|T\|_\rho]}$. Hence $(V, p)[\sigma] \in \bigcup_{v \in \|\tau\|} \|A\|_{\rho[x^\tau \leftarrow v]} \subseteq \|\exists x^\tau. A\|_\rho$.
- (\exists^τ) We now prove the case where t is not a value (hence, the formula is positive). Suppose $\vdash t : P[T/x^\tau] \mid \Gamma$ is p -adequate for some $T : \tau$. We want to show that $\vdash t : \exists x^\tau. P \mid \Gamma$ is p -adequate. Let ρ be a total valuation and $\sigma \Vdash \Gamma[\rho]$. We can suppose that x^τ does not appear in Γ (if it does, then we can rename it in $\exists x^\tau. P$). Because of p -adequacy of the premise, we have $(t, p)[\sigma] \in \|P[T/x^\tau]\|_\rho^{\perp\perp}$. But by Property 31, we have $\|P[T/x^\tau]\|_\rho^{\perp\perp} = \|P\|_{\rho[x^\tau \leftarrow \|T\|_\rho]}^{\perp\perp}$. Hence $(t, p)[\sigma] \in \bigcup_{v \in \|\tau\|} \|P\|_{\rho[x^\tau \leftarrow v]}^{\perp\perp} \subseteq \|\exists x^\tau. P\|_\rho^{\perp\perp}$.

If π is a typing derivation, we define $\mathbf{M}[\pi]$ as the element of \mathcal{M} obtained by composing the $\mathbf{M}[R]$ of each rule appearing in π in the obvious way. Hence, if π is a typing derivation of $\mathbf{MAL}\omega$, then Theorem 42 says that its conclusion is $\mathbf{M}[\pi]$ -adequate.

Remark 43. To prove the adequacy theorem, we crucially rely on the saturation properties of $\perp\!\!\!\perp$. This is where we really need to have a saturated quantitative pole.

3.7. Non quantitative Krivine's classical realizability

In a particular case of our general quantitative classical realizability, it is possible to validate a contraction rule (hence dropping the linearity constraint) and recover the non quantitative version of Krivine's classical realizability [23], which we call *simple realizability*. It will be used in Section 5 to state the forcing decomposition of quantitative realizability.

Definition 44. We note \mathcal{M}_0 the only quantitative monoid (without unit) whose underlying set is $\{0\}$ equipped with the usual addition on natural numbers.

Suppose we have a *non quantitative pole* $\perp\!\!\!\perp \subseteq \mathcal{C}$, that is a set of commands such that:

$$c' \in \perp\!\!\!\perp \text{ and } c \rightarrow_0 c' \text{ implies } c \in \perp\!\!\!\perp$$

Then we define a *quantitative extension* of $\perp\!\!\!\perp$:

$$\perp\!\!\!\perp_0 = \{ (c, 0) \mid c \in \perp\!\!\!\perp \}$$

Property 45. *The structure $(\mathcal{M}_0, \perp\!\!\!\perp_0, 0)$ is a saturated quantitative pole.*

PROOF. \mathcal{M}_0 is clearly a quantitative monoid. Moreover, if $c \rightarrow_0 c'$ and $(c', 0) \in \perp\!\!\!\perp_0$, we have $c' \in \perp\!\!\!\perp$ hence $c \in \perp\!\!\!\perp$. That implies $(c, 0) \in \perp\!\!\!\perp_0$, which proves both the \rightarrow_β and \rightarrow_μ -saturation properties. Finally, the \leq -saturation is immediate, since \mathcal{M}_0 is a singleton.

This quantitative pole induces an interpretation function $\|\cdot\|_\rho$ and a realizability relation $\Vdash_{\perp\!\!\!\perp_0}^\rho$. We define the simple realizability relation \Vdash as:

$$t \Vdash^\rho T \iff (t, 0) \Vdash_{\perp\!\!\!\perp_0}^\rho T$$

If we add the following contraction rule to $\mathbf{MAL}\omega$, we obtain a formulation of $\mathbf{PA}\omega$ (higher-order Peano arithmetic):

$$\frac{c : (\vdash \kappa_1 : A, \kappa_2 : A, \Gamma)}{c[\kappa/\kappa_1, \kappa/\kappa_2] : (\vdash \kappa : A, \Gamma)} (C)$$

Although it does not hold in general, when we use $\perp\!\!\!\perp_0$ and pose $\mathbf{M}[C] = x \mapsto x$, the contraction rule is adequate:

Property 46. *The rule C is $\mathbf{M}[C]$ -adequate.*

PROOF. Suppose $c : (\vdash \kappa_1 : A, \kappa_2 : A, \Gamma)$ is 0-adequate. Let $\rho \Vdash \Gamma, A$ and $\sigma \Vdash (\Gamma, \kappa : A)[\rho]$. Then $\sigma = \sigma', \kappa \mapsto (u, q)$. \mathcal{M}_0 is a singleton so $q = 0$. If we pose $\tau = \sigma', \kappa \mapsto (u, 0), \kappa_2 \mapsto (u, 0)$ then $\tau \Vdash (\Gamma, \kappa_1 : A, \kappa_2 : A)[\rho]$. Hence, by adequacy of the premise, $(c, 0)[\tau] \in \perp_0$. Since $0 + 0 = 0$, we also have $(c[\kappa/\kappa_1, \kappa/\kappa_2], 0)[\sigma] = (c, 0)[\tau]$. Hence the conclusion is 0-adequate.

Hence, as a corollary of Theorem 42 and Property 46, we recover an adequacy theorem for $\mathbf{PA}\omega$.

Theorem 47. *All rules R of $\mathbf{PA}\omega$ are $\mathbf{M}[R]$ -adequate.*

Remark 48. In the case of the non-quantitative realizability, the notions of p -adequate judgments and f -adequate rules can be simplified. We will say that a judgment is *adequate* if it is 0-adequate, and a rule is *adequate* if it is $(x \mapsto x)$ -adequate.

The following remark show that this version of the contraction rule is not $x \mapsto x$ -adequate in general.

Remark 49. When considering the general quantitative framework, this version of the contraction rule C is never f -adequate for $f = x \mapsto x$ as soon as the quantitative pole meets the following conditions:

- The quantitative monoid has a unit $\mathbf{1}$ (for example the integers monoid)
- There is a command c such that $c : (\vdash_{\mathbf{MAL}\omega} x : X, y : X, \Gamma)$, a valuation $\rho \Vdash X, \Gamma$, a substitution $\sigma \Vdash \Gamma[\rho]$ and such that there is some $(u, q) \in \rho(X)$ such that $(c[u/x, u/y], p + q)[\sigma] \notin \perp$.

PROOF. Suppose the rule C is f -adequate for $f = x \mapsto x$. By Theorem 42 we know that the judgment $c : (\vdash_{\mathbf{MAL}\omega} x : X, y : X, \Gamma)$ is p -adequate for some p . Hence, we have $(c[u/x, u/y], p + 2q)[\sigma] \in \perp$ by p -adequacy of the typing judgment and because $\sigma \Vdash \Gamma[\rho]$. Moreover, since C is adequate, its conclusion must be $f(p)$ -adequate. So we have $(c[u/x, u/y], p + q)[\sigma] \in \perp$, which is contradictory with the assumptions.

3.8. Quantitative reducibility candidates model

In this subsection, we build a particular class of quantitative realizability models. By applying Theorem 42 on these models, we can prove a linear time termination property of $\mathbf{MAL}\omega$ programs. Later, this result will be extended to more sophisticated systems that also enjoy bounded-time termination properties. The construction relies on the definition of a quantitative extension of the well-known *reducibility candidates* (defined by orthogonality, as in [19, 7]), which we call *quantitative reducibility candidates*.

In the rest of this subsection, the quantitative monoid and the pole are such that:

- The quantitative monoid is any quantitative monoid with unit $\mathcal{M} = (M, +, \mathbf{0}, \|\cdot\|, \mathbf{1})$.
- The quantitative pole is the structure $(\mathcal{M}, \perp, p_\beta)$ generalizing the one described in Example 21:
 - $\perp = \{ (c, p) \mid \mathbf{Time}^{\rightarrow\beta}(c) \text{ is defined and } \mathbf{Time}^{\rightarrow\beta}(c) \leq \|p\| \}$
 - $p_\beta = \mathbf{1}$

We now use the fact that our syntax can be extended: we suppose that our two instruction sets \mathcal{K}_+ and \mathcal{K}_- contain respectively the constants written \mathfrak{X}_+ and \mathfrak{X}_- . These two constants play the same role as free variables in the usual reducibility candidates argument. The only relevant properties of these new constants are:

Property 50. *If $V_+ \in \mathcal{T}_+^0$, $t_- \in \mathcal{T}_-^0$ and $\mathfrak{X}_1, \mathfrak{X}_2 \in \{\mathfrak{X}_+, \mathfrak{X}_-\}$, then*

1. $\langle V_+ \mid \mathfrak{X}_- \rangle \rightarrow_0$
2. $\mathbf{Time}^{\rightarrow\beta}(\langle t_- \mid \mathfrak{X}_+ \rangle) \leq \mathbf{Time}^{\rightarrow\beta}(\langle t_- \mid (\mathfrak{X}_1, \mathfrak{X}_2) \rangle)$
3. $\mathbf{Time}^{\rightarrow\beta}(\langle t_- \mid \mathfrak{X}_+ \rangle) \leq \mathbf{Time}^{\rightarrow\beta}(\langle t_- \mid \{\mathfrak{X}_1\} \rangle)$

PROOF. All these properties are immediate.

Definition 51 (Quantitative reducibility candidates). The set of *positive quantitative reducibility candidates*, denoted by \mathcal{D}_{can}^\oplus , is the set of elements $X \in \mathcal{P}(\mathcal{T}_+^0 \cap \mathbb{V} \times \mathcal{M})$ such that:

1. $(X^{\perp\perp})_{\mathbb{V}} = X$
2. $(\mathfrak{X}_+, \mathbf{0}) \in X^{\perp\perp}$
3. $X^{\perp\perp} \subseteq \{(\mathfrak{X}_-, \mathbf{0})\}^\perp$

The set $\mathcal{D}_{can}^\ominus = \{ X^\perp \mid X \in \mathcal{D}_{can}^\oplus \}$ is the set of *negative quantitative reducibility candidates*. The set of *quantitative reducibility candidates* \mathcal{D}_{can} is the set $\mathcal{D}_{can}^\ominus \cup \mathcal{D}_{can}^\oplus$.

The following lemmas are used to prove that the set \mathcal{D}_{can}^\oplus can be used as a positive propositional domain. We have to check every closure condition of Definition 27

Lemma 52. *Whenever $X, Y \in \mathcal{D}_{can}$, then $X \otimes Y \in \mathcal{D}_{can}^\oplus$.*

PROOF. • We want to show that $(\mathfrak{X}_+, \mathbf{0}) \in (X \otimes Y)^{\perp\perp}$. Let's take some $(t_-, p) \in (X \otimes Y)^\perp$. By Lemma 34, we have also $(t_-, p) \in (X^{\perp\perp} \otimes Y^{\perp\perp})^\perp$. Depending of the polarity of X and Y , we know that $\mathfrak{X}_1 \in X^{\perp\perp}$ and $\mathfrak{X}_2 \in Y^{\perp\perp}$ for some $\mathfrak{X}_1, \mathfrak{X}_2 \in \{\mathfrak{X}_+, \mathfrak{X}_-\}$. So $(\langle t_- \mid (\mathfrak{X}_1, \mathfrak{X}_2) \rangle, p) \in \perp$. But by Property 50,

$$\begin{aligned} \mathbf{Time}^{\rightarrow\beta}(\langle t_- \mid \mathfrak{X}_+ \rangle) &\leq \mathbf{Time}^{\rightarrow\beta}(\langle t_- \mid (\mathfrak{X}_1, \mathfrak{X}_2) \rangle) \\ &\leq \|p\| \end{aligned}$$

we can conclude that $(t_-, p) \perp (\mathfrak{X}_+, \mathbf{0})$. So $(\mathfrak{X}_+, \mathbf{0}) \in (X \otimes Y)^{\perp\perp}$.

- We now need to show that $X \otimes Y \subseteq \{(\mathbf{x}_-, \mathbf{0})\}^\perp$. We know that $X \otimes Y$ only contains values, since $X, Y \in \mathcal{D}_{can}$. But now, it is easy to see that if $(V_+, p) \in X \otimes Y$, then immediately $\langle V_+ | \mathbf{x}_- \rangle$ does not reduce for \rightarrow_0 and so $(\langle V_+ | \mathbf{x}_- \rangle, p) \in \perp$.

Lemma 53. *Whenever $X \in \mathcal{D}_{can}$, then $(\downarrow X)^{\perp\perp} \in \mathcal{D}_{can}^\oplus$.*

PROOF. • We want to show that $(\mathbf{x}_+, \mathbf{0}) \in (\downarrow X)^{\perp\perp}$. Let's take some $(t_-, p) \in (\downarrow X)^\perp$. By Lemma 34, we know that $(t_-, p) \in (\downarrow X^{\perp\perp})^\perp$. But, depending of the polarity of X , we know that $(\langle t_- | \mathbf{x} \rangle, p) \in \perp$ with $\mathbf{x} \in \{\mathbf{x}_+, \mathbf{x}_-\}$. In any case, because

$$\begin{aligned} \mathbf{Time}^{\rightarrow\beta}(\langle t_- | \mathbf{x}_+ \rangle) &\leq \mathbf{Time}^{\rightarrow\beta}(\langle t_- | \{\mathbf{x}\} \rangle) \\ &\leq \|p\| \end{aligned}$$

, we can conclude that $(t_-, p) \perp (\mathbf{x}_+, \mathbf{0})$. So $(\mathbf{x}_+, \mathbf{0}) \in (\downarrow X)^{\perp\perp}$.

- We now need to show that $(\downarrow X)^{\perp\perp} \subseteq \{(\mathbf{x}_-, \mathbf{0})\}^\perp$. By orthogonality properties, it suffices to show that $\downarrow X \subseteq \{(\mathbf{x}_-, \mathbf{0})\}^\perp$. Since X contains only values, so does $\downarrow X$. But it is immediate that for any $(V_+, p) \in \downarrow X$, $\langle V_+ | \mathbf{x}_- \rangle$ does not reduce for \rightarrow_0 and so $(\langle V_+ | \mathbf{x}_- \rangle, p) \in \perp$.

Lemma 54. *Suppose $\emptyset \neq D \subseteq \mathcal{D}_{can}^\oplus$, then $\bigcap_{X \in D} X \in \mathcal{D}_{can}^\oplus$.*

PROOF. Suppose $\emptyset \neq D \subseteq \mathcal{D}_{can}^\oplus$.

- By hypothesis, for each $X \in D$ we have $(\mathbf{x}_+, \mathbf{0}) \in X^{\perp\perp}$. So it is immediate that $(\mathbf{x}_+, \mathbf{0}) \in \bigcap_{X \in D} X^{\perp\perp} = (\bigcap_{X \in D} X)^{\perp\perp}$ by Lemma 34.
- Because each $X \in D$ is such that $X \subseteq X^{\perp\perp} \subseteq \{(\mathbf{x}_-, \mathbf{0})\}^\perp$, it is clear that $\bigcap_{X \in D} X \subseteq \{(\mathbf{x}_-, \mathbf{0})\}^\perp$ (since D is not empty and contains only non-empty sets).

Lemma 55. *Suppose $\emptyset \neq D \subseteq \mathcal{D}_{can}^\oplus$, then $\bigcup_{X \in D} X \in \mathcal{D}_{can}^\oplus$.*

PROOF. Suppose $\emptyset \neq D \subseteq \mathcal{D}_{can}^\oplus$.

- By hypothesis, for each $X \in D$ we have $(\mathbf{x}_+, \mathbf{0}) \in X^{\perp\perp}$. Since D is not empty we have $(\mathbf{x}_+, \mathbf{0}) \in \bigcup_{X \in D} X^{\perp\perp} \subseteq (\bigcup_{X \in D} X^{\perp\perp})^{\perp\perp}$. But this is equal to $(\bigcup_{X \in D} X)^{\perp\perp}$ by Lemma 34.
- Because each $X \in D$ is such that $X \subseteq X^{\perp\perp} \subseteq \{(\mathbf{x}_-, \mathbf{0})\}^\perp$, it is clear that $\bigcup_{X \in D} X \subseteq \{(\mathbf{x}_-, \mathbf{0})\}^\perp$.

These four lemmas permit us to conclude that if we choose the set \mathcal{D}_{can}^\oplus as the positive propositional domain, then for each formula A and each valuation $\rho \Vdash A$, $\|A\|_\rho$ is a quantitative reducibility candidate.

Linear time termination — As an example, we show how to use the adequacy theorem on the quantitative reducibility candidates model to prove complexity properties of terms typable in $\mathbf{MAL}\omega$. To do this, we need to choose a concrete quantitative monoid with unit: we take the natural numbers quantitative monoid defined in Example 15. We obtain the following theorem:

Theorem 56. *If $c : (\vdash \Gamma)$, then c normalizes for \rightarrow_0 using at most $|c| \rightarrow_\beta$ -steps.*

PROOF. Suppose that π is a proof of $c : (\vdash \kappa_1 : A_1, \dots, \kappa_n : A_n)$. We set:

- The quantitative monoid of integers \mathbb{N} .
- The quantitative pole $\perp\!\!\!\perp_{\mathbf{Time}}$.

Let ρ be a total valuation such that $\rho(x^{o^+}) = \{(\mathfrak{X}_+, \mathbf{0})\} \in \mathcal{D}_{can}^\oplus$ (such a valuation exists). By Theorem 42, we know that for all $(V_i, q_i) \in \|A_i\|_\rho$, we have

$$(c[V_1/\kappa_1, \dots, V_n/\kappa_n], \mathbf{M}[\pi] + \sum_i q_i) \in \perp\!\!\!\perp$$

By Lemma 35, for every $(W_i, p_i) \in \|A_i\|_\rho^{\perp\!\!\!\perp} \cap \mathbb{V}$,

$$(c[W_1/\kappa_1, \dots, X_n/\kappa_n], \mathbf{M}[\pi] + \sum_i p_i) \in \perp\!\!\!\perp$$

Since \mathcal{D}_{can}^\oplus is a positive propositional domain, we know that $(\mathfrak{X}_i, \mathbf{0}) \in \|A_i\|_\rho^{\perp\!\!\!\perp}$, where $\mathfrak{X}_i \in \{\mathfrak{X}_+, \mathfrak{X}_-\}$ depending of the polarity of A_i . It implies that

$$(c[\mathfrak{X}_1/\kappa_1, \dots, \mathfrak{X}_n/\kappa_n], \mathbf{M}[\pi]) \in \perp\!\!\!\perp$$

Hence,

$$\begin{aligned} \mathbf{Time}^{\rightarrow_\beta}(c) &= \mathbf{Time}^{\rightarrow_\beta}(c[\mathfrak{X}_1/\kappa_1, \dots, \mathfrak{X}_n/\kappa_n]) \\ &\leq \|\mathbf{M}[\pi]\| \end{aligned}$$

But, it is easy to see that $\mathbf{M}[\pi] \leq |c|$. Hence $\mathbf{Time}^{\rightarrow_\beta}(c) \leq |c|$.

4. Extending the model: Soft Affine Logic

So far, we have only treated the multiplicative fragment of $\mathbf{PA}\omega$. In this section, we show how to extend the realizability interpretation to more substantial fragments. We take the particular example of Soft Affine Logic (augmented with higher-order quantifiers and arithmetical operations, and noted $\mathbf{SAL}\omega$). To do this, we need to find a suitable quantitative monoid: this is done by turning the *soft resource monoid* defined in [18] into a quantitative monoid. We then extend the adequacy theorem and the construction of quantitative reducibility candidates to this new system, finally we prove a polynomial bounded-time normalization property. The same methodology can be applied without any trouble to all the systems handled in [18].

4.1. Soft affine logic

Soft affine logic [2], is a simple extension of multiplicative linear logic by mean of weak exponentials. It ensures a polystep normalization property of programs typable in this system.

We now suppose that the set \mathcal{K}_+ of positive instructions contains a term $!V$ for each value V . We also suppose that \mathcal{K}_- contains a term $\mu!(\kappa).c$ for each command c and variable κ . We also suppose that the reduction relation \rightarrow contains the following binary relation $\rightarrow_!$:

$$\langle \mu!(\kappa).c \mid !V \rangle \rightarrow_! c[V/\kappa]$$

Suppose $\{x_1, \dots, x_k\}$ is a set of positive variables, t is a term and κ is a fresh variable of the same polarity as t . Then we define respectively a command $!_{\{x_1, \dots, x_k\}}^\kappa t$ and a term $!_{\{x_1, \dots, x_k\}} t$ inductively as follows:

$$\begin{aligned} !_{\emptyset}^\kappa t &= \langle !t \mid \kappa \rangle \\ !_{S \cup \{x\}}^\kappa t &= \langle \mu!(x).(!_S^\kappa t) \mid x \rangle \\ !_S t &= \mu\kappa.(!_S^\kappa t) \end{aligned}$$

For example, with the variables x_1, \dots, x_k , we have

$$!_{\{x_1, \dots, x_k\}} t = \mu\kappa. \langle \mu!(x_1). \langle \mu!(x_2). \langle \dots \langle !t \mid \kappa \rangle \mid \dots \rangle \dots \mid x_2 \rangle \mid x_1 \rangle$$

Property 57. Suppose x_1, \dots, x_k are positive variables and t is a term whose free variables are included in $\{x_1, \dots, x_k\}$. Suppose moreover that u_1, \dots, u_k are positive values and V is a closed value of the opposite polarity as t 's polarity. Then we have

$$\langle !_{\{x_1, \dots, x_k\}} t \mid V \rangle [!u_1/x_1, \dots, !u_k/x_k] \rightarrow_\mu \rightarrow_!^k \langle !t \mid V \rangle [u_1/x_1, \dots, u_k/x_k]$$

Remark 58. The construction $!_{\{x_1, \dots, x_k\}}$ is here to mimic the functorial $!$ box of **SAL** proofnets. The last property is then the counterpart of the reduction resulting of the interaction between several functorial boxes.

We add two new formula constructors $?A$ and $!A$.

$$A, B, T, U ::= \dots \mid !A \mid ?A$$

The formulas $!A$ and $?A$ are respectively positive and negative formulas, in accordance with the following two new constructor typing rules:

$$\frac{A : o}{!A : o^+} \quad \frac{A : o}{?A : o^-}$$

Typing rules of **SAL** ω are then obtained by extending the rules presented in Subsection 2 with the following ones:

$$\frac{\vdash V : A \mid x_1 : N_1, \dots, x_k : N_k}{\vdash !_{\{x_1, \dots, x_k\}} V : !A \mid x_1 : ?N_1, \dots, x_k : ?N_k} (!_k) \quad \frac{c : (\vdash \kappa_1 : A, \dots, \kappa_n : A, \Gamma)}{\vdash \mu!(\kappa).c[\kappa/\kappa_1, \dots, \kappa/\kappa_n] : ?A \mid \Gamma} (M_n)$$

Remarks 59.

1. The choice of having only negative formulas in the context of the $!$ rule is not restrictive, since we can always use the \uparrow rule to obtain such a context. But doing so allows not to care about the polarity of variables.
2. The multiplex rule (M_n) and the promotion $(!_n)$ rules are in fact typing schemes, that is one rule for each integer $n \in \mathbb{N}$. In the case of the promotion rule, this in fact accounts for the fact that functorial promotion is a cluster rule consisting of n derelictions and one usual promotion.
3. Notice that the multiplex rule, in the $n = 1$ case is exactly what we usually call the *dereliction rule*. One could think that it is possible to decompose the multiplex rule in two more elementary rules: the dereliction rule and the usual contraction. However, this would lead to typable programs that can calculate functions that are not computable in polynomial time.

If π is a typing derivation in **SAL** ω , then we define its *depth* $\delta(\pi)$ as the maximum number of nested $(!)$ rules appearing in it. In the rest of this paper, we will use the symbol $\vdash_{\text{SAL}\omega}$ instead of \vdash when we talk about typability in this new type system.

4.2. Soft monoid

In order to obtain a model where those rules are adequate, we need a richer structure than the quantitative monoid. This structure is given by the notion of *soft exponential*.

Definition 60. Let $\mathcal{M} = (M, +, 0, \leq, \|\cdot\|)$ be a quantitative monoid. Then a *soft exponential* on \mathcal{M} is given by a family $(r_n)_{n \in \mathbb{N}}$ of elements of \mathcal{M} and an operation $! : \mathcal{M} \rightarrow \mathcal{M}$ that satisfy the following properties:

- For all $p, q \in \mathcal{M}$, we have $!p + !q \leq !(p + q)$.
- For all $p \in \mathcal{M}$ and $n \in \mathbb{N}$, we have $n.p \leq !p + r_n$.

We now give a concrete example of a quantitative monoid with unit and soft exponential. This monoid is obtained from the *soft resource monoid* described in [18].

Definition 61. The *soft monoid* is the structure $\mathcal{M}_s = (M_s, +_s, \mathbf{0}_s, \mathbf{1}_s, \leq_s, \|\cdot\|_s)$ where

- M_s is the set of pairs (n, f) where $n \in \mathbb{N}$ and $f \in \mathbb{N}[X]$ is a polynomial with integer coefficients.
- $(n, f) +_s (m, g) = (\max(n, m), f + g)$ where $\max(n, m)$ is the maximum of n and m .

- $\mathbf{0}_s = (0, x \mapsto 0)$ and $\mathbf{1}_s = (0, 1)$.
- $(n, f) \leq_s (m, g)$ iff $n \leq m$ and $\forall x \geq m, f(x) \leq g(x)$ and $(g - f)(x) \leq (g - f)(y)$ for $m \leq x \leq y$.
- $\|(n, f)\|_s = f(n)$.

Property 62. \mathcal{M}_s is a quantitative monoid with unit.

PROOF. • It is clear that $(\mathcal{M}_s, +_s, \mathbf{0}_s, \leq_s)$ is a preordered commutative monoid.

- Let (n, f) and (m, g) be two elements of M_s . We have

$$\begin{aligned}
\|(n, f)\| + \|(m, g)\| &= f(n) + g(m) \\
&\leq f(\max(n, m)) + g(\max(n, m)) \\
&= (f + g)(\max(n, m)) \\
&= \|(\max(n, m), f + g)\| \\
&= \|(n, f) +_s (m, g)\|
\end{aligned}$$

- Suppose $(n, f) \leq_s (m, g)$. It means that $n \leq m$ and $\forall x \in \mathbb{N}$ such that $x \geq m, f(x) \leq g(x)$. Hence, we have

$$\|(n, f)\| = f(n) \leq f(m) \leq g(m) = \|(m, g)\|$$

- Finally $\mathbf{1}$ is a unit, since $\|\mathbf{1}\|_s = 1$.

We moreover define the operation $! : M_s \rightarrow M_s$ as $!(n, f) = (n, f^+)$ where $f^+(X) = (X + 1)f(X)$. This operation enjoys various properties.

Property 63. The pair $(!, \{(n, 0)\}_{n \in \mathbb{N}})$ is a soft exponential.

PROOF. Here, we pose $p = (n, f)$ and $q = (m, g)$.

(i) We have

$$\begin{aligned}
!(p +_s q) &= (\max(n, m), (f + g)^+) \\
&= (\max(n, m), (X + 1)(f + g)) \\
&= (\max(n, m), (X + 1)f + (X + 1)g) \\
&= (\max(n, m), f^+ + g^+) \\
&= !p +_s !q
\end{aligned}$$

(ii) We have

$$\begin{aligned}
k.p &= (n, k.f) \\
&\leq_s (\max(n, k), (X + 1)f) \\
&= (\max(n, k), f^+) \\
&= !p +_s (k, 0)
\end{aligned}$$

(iii) Immediate.

Properties (i) and (ii) are crucial to obtain respectively monoidality of $!$ and the multiplexing rule (hence to prove adequacy).

4.3. Interpretation of **SAL** ω

We now extend the realizability interpretation defined on the multiplicative fragment to the exponentials. We suppose that:

- \rightarrow is an evaluation relation that contains $\rightarrow_!$.
- \mathcal{M} is a quantitative monoid with a soft exponential $(!, (r_n)_{n \in \mathbb{N}})$.
- $(\mathcal{M}, \perp, p_\beta)$ is a quantitative pole which is moreover $\rightarrow_!$ -saturated:

For every $(c, p) \in \perp$, if $c' \rightarrow_! c$ then $(c', p + p_\beta) \in \perp$

We introduce a new unary operation $!$ on sets of bounded terms:

$$!X = \{ (!V, !p) \mid (V, p) \in (X^{\perp\perp})_{\mathbb{V}} \times \mathcal{M} \}$$

and we extend the interpretation of formulas as follows:

$$\begin{aligned} \|!A\|_\rho &= !\|A\|_\rho \\ \|\?A\|_\rho &= (!\|A^\perp\|_\rho)^\perp \end{aligned}$$

As in Subsection 3.6, to state the adequacy theorem we need to associate a function to the two new rules:

$$\begin{aligned} \mathbf{M}[!_n] &= x \mapsto !x + n.p_\beta \\ \mathbf{M}[M_n] &= x \mapsto x + p_\beta + r_n \end{aligned}$$

Theorem 64. *All rules R of **SAL** ω are $\mathbf{M}[R]$ -adequate.*

PROOF. • For all the multiplicative part, the proof is the same as the one of the **MAL** ω adequacy theorem.

- $(!)$: Suppose $\vdash V : A \mid \Gamma$ is p -adequate. Let ρ be a total valuation and $\sigma \Vdash ?\Gamma[\rho]$. We know that for each $x_i : N_i \in \Gamma$, we have $\sigma(x_i) = (!V_i, !q_i)$ where $(V_i, q_i) \in \|N_i^\perp\|_\rho$. If we pose $\sigma' = [x_1 \leftarrow (V_1, q_1), \dots, x_k \leftarrow (V_k, q_k)]$, we have clearly $\sigma' \Vdash \Gamma[\rho]$. By hypothesis $(V, p)[\sigma'] \in \overline{\|A\|_\rho}$. Hence, $(V, p)[\sigma'] \in \|A\|_\rho^{\perp\perp} \cap \mathbb{V}$ and finally $(!V[V_1/x_1, \dots, V_k/x_k], !(p + q_1 + \dots + q_k)) \in \|!A\|_\rho$. Because $!$ is a soft exponential on \mathcal{M} , and by \leq -saturation, we obtain $(!V[V_1/x_1, \dots, V_k/x_k], !p + !q_1 + \dots + !q_k) \in \|!A\|_\rho$. Finally, we know by Property 57, \rightarrow_β and \rightarrow_μ -saturation that

$$(!_{\{x_1, \dots, x_k\}} V[V_1/x_1, \dots, V_k/x_k], !p + k.p_\beta + !q_1 + \dots + !q_k) \in \overline{\|!A\|_\rho}$$

which can be rewritten as

$$(!_{\{x_1, \dots, x_k\}} V, !p + k.p_\beta)[\sigma] \in \overline{\|!A\|_\rho}$$

Hence $\vdash V : !A \mid \Gamma$ is $!p$ -adequate and so the $!_k$ rule is $(x \mapsto !x + k.p_\beta)$ -adequate.

- (*Mplex_n*): Suppose $c : (\vdash \kappa_1 : A, \dots, \kappa_n : A, \Gamma)$ is p -adequate, let ρ be a total valuation and $\sigma \Vdash \Gamma$. We want to show that

$$(\mu!(\kappa).c[\kappa/\kappa_1, \dots, \kappa/\kappa_n], p + p_\beta + r_n)[\sigma] \in \|\?A\|_\rho = \|!A^\perp\|_\rho^\perp$$

First, notice that if $(V, q) \in \|A\|_\rho$, if we pose $\sigma' = \sigma[\kappa_1 \leftarrow V, \dots, \kappa_n \leftarrow V]$, we clearly have $\sigma' \Vdash (\kappa_1 : A, \dots, \kappa_n : A, \Gamma)[\rho]$. By p -adequacy of the hypothesis, we obtain $(c[V/\kappa_1, \dots, V/\kappa_n], p + n.q)[\sigma] \in \perp$.

Now, let $(V, q) \in \|A\|_\rho^{\perp\perp} \cap \mathbb{V}$. We want to show that

$$\mu!(\kappa).c[\kappa/\kappa_1, \dots, \kappa/\kappa_n], p + p_\beta)[\sigma] \perp (!V, !q)$$

which will prove that the conclusion is $(p + p_\beta)$ -adequate. But we know that $\langle \mu!(\kappa).c[\kappa/\kappa_1, \dots, \kappa/\kappa_n] \mid !V \rangle \rightarrow_0 c[V/\kappa_1, \dots, V/\kappa_n]$. But by the previous point, combined n times with Lemma 35, we obtain

$$\forall (V', q') \in X^{\perp\perp}, (c[V'/\kappa_1, \dots, V'/\kappa_n], p + n.q')[\sigma] \in \perp$$

Hence $(c[V/\kappa_1, \dots, V/\kappa_n], p + n.q) \in \perp$. By \leq -saturation of \perp and because $!$ is a soft exponential, we have $(c[V/\kappa_1, \dots, V/\kappa_n], p + !q + r_n) \in \perp$. By \rightarrow_β -saturation we finally obtain

$$(\mu!(\kappa).c[\kappa/\kappa_1, \dots, \kappa/\kappa_n], p + p_\beta + r_n)[\sigma] \in \|\?A\|_\rho$$

Remark 65. In this proof, the lemma 35 is crucial to show adequacy of the multiplex rule. The situation would be similar for any system containing modality-rules that change the whole context.

4.4. Polynomial bounded time termination

We now prove the polynomial bounded time termination of **SAL** ω , by extending the technique of quantitative reducibility candidates.

To obtain a bounded normalization theorem, we need to check that the construction of the quantitative reducibility candidates is still valid. The definitions remain the same as those of Subsection 3.8, except for the definition of \perp :

$$\perp = \{ (c, p) \mid \mathbf{Time}^{(\rightarrow_\beta \cup \rightarrow_!)}(c) \text{ is defined and is bounded by } \|p\| \}$$

The only new property we need is the following one:

Lemma 66. *If $X \in \mathcal{D}_{can}$ then $!X \in \mathcal{D}_{can}^\oplus$.*

PROOF. Without any loss of generality, let's suppose X is positive.

- Let $(t, p) \in (!X)^\perp$. We have to show that $(\mathfrak{X}_+, \mathbf{0}) \perp (t, p)$. But we know that $(\mathfrak{X}_+, \mathbf{0}) \in X^{\perp\perp}$. Hence, $(!\mathfrak{X}_+, !\mathbf{0}) \in !X$. By Property 63, $!\mathbf{0} = \mathbf{0}$. We then have $(\langle t \mid !\mathfrak{X}_+ \rangle, p + \mathbf{0}) \in \perp$. But we know that

$$\begin{aligned} \mathbf{Time}^{(\rightarrow_\beta \cup \rightarrow_!)}(\langle \mathfrak{X}_+ \mid t \rangle) &\leq \mathbf{Time}^{(\rightarrow_\beta \cup \rightarrow_!)}(\langle !\mathfrak{X}_+ \mid t \rangle) \\ &\leq \|p\| \end{aligned}$$

Hence $(\mathfrak{X}_+, \mathbf{0}) \in (!X)^{\perp\perp}$.

- Let $(V, p) \in !X$, because V is a positive value, it is immediate that $\langle V \mid \mathfrak{X}_- \rangle$ does not reduce for $\rightarrow_0 \cup \rightarrow_!$. Hence $!X \subseteq \{(\mathfrak{X}_-, \mathbf{0})\}^\perp$ and so $(!X)^{\perp\perp} \subseteq \{(\mathfrak{X}_-, \mathbf{0})\}^\perp$.

By instantiating the monoid with the soft monoid, we can derive a polystep normalization property of terms typable in **SAL** ω which extends the linear time normalization property of Subsection 3.8.

Theorem 67. *There exists a family $(P_k)_{k \in \mathbb{N}}$ of polynomials on \mathbb{N} such that if π is a proof of $\vdash_{\mathbf{SAL}\omega} t : A \mid$, then t normalizes in at most $P_{\delta(\pi)}(|t|)$ reduction steps.*

PROOF. The proof consists essentially to remark that in the definition of $\mathbf{M}[\pi]$, the only rule that makes the degree of the polynomial of $\mathbf{M}[\pi]$ rise is the $(!)$ rule. The other rules cause only the linear part of $\mathbf{M}[\pi]$ to grow.

5. A forcing decomposition

In this section, we exhibit a connection between our quantitative extension of classical realizability and certain forcing interpretations. We precisely show that by composing non-quantitative classical realizability with a notion of forcing for **MAL**, we obtain an instance of quantitative realizability. We finally show that quantitative reducibility candidates are a special case of this construction. We proceed with the following methodology:

1. We define the notion of *linear forcing structure*, a variation of the notion of forcing structure already defined in [14].
2. We introduce a *forcing translation*, that is the formalization of a class of forcing model of **MAL** inside **MAL** ω . The result is a relation $p \Vdash_f A$, parametrized by a choice of linear forcing structure.
3. We describe a new machine: the *countdown machine*. It is based on the same term syntax as **L** $_{foc}$, but with a different notion of command and different reduction rules. This machine induces a new class of non-quantitative realizability relations for **MAL** ω , parametrized by a set \perp^\bullet and denoted $t \Vdash^\bullet A$.

4. We show that for a *particular* instance of linear forcing structure and for *every* choice of \perp^\bullet , there exists a *quantitative pole* (in the sense of Section 3) \perp° such that the associated realizability relation \Vdash° satisfies for every **MAL** formula A :

$$(t, p) \Vdash^\circ A \iff t \Vdash^\bullet (p \Vdash_f A)$$

That means composing \Vdash_f and \Vdash^\bullet yields a quantitative model of **MAL**

5. Finally, we show that quantitative reducibility candidates of Subsection 3.8, restricted to **MAL**, can be seen as the result of such a composition.

This methodology could be used to study forcing translations of **MAL** ω , but we believe it is simpler to explain it with **MAL**. In the last subsection we explain how it could be extended to the whole system **MAL** ω .

5.1. Preliminaries

We define some concepts and notations required to define the forcing translation and establish the associated results.

Multiplicative Affine Logic — **MAL** (Multiplicative Affine Logic) is the affine, second-order fragment of **MAL** ω . The following grammar defines **MAL** formulas:

$$\begin{aligned} A, B &::= P \mid N \\ P &::= X \mid A \otimes B \mid \downarrow A \\ N &::= X^\perp \mid A \wp B \mid \uparrow A \end{aligned}$$

We suppose that to each **MAL** variable X we associate a **MAL** ω variable X°^+} . Then any **MAL** formula A can be seen as a **MAL** ω constructor A^ω of kind o° (with $\circ \in \{+, -\}$), defined as follows:

$$\begin{aligned} (X)^\omega &= X^{\circ^+} \\ (X^\perp)^\omega &= (X^{\circ^+})^\perp \\ (A \otimes B)^\omega &= (A)^\omega \otimes (B)^\omega \\ (A \wp B)^\omega &= (A)^\omega \wp (B)^\omega \\ (\downarrow A)^\omega &= \downarrow (A)^\omega \\ (\uparrow A)^\omega &= \uparrow (A)^\omega \end{aligned}$$

We will abusively use the same notation A for both the **MAL** formula A and its associated **MAL** ω constructor $(A)^\omega$. We will also use the notation $A \multimap B = A^\perp \wp B$, that is the call-by-name linear implication.

Realizability relations — In the rest of this section, we will manipulate several realizability relations. We will in particular consider a quantitative realizability relation (that will be denoted by \Vdash° in further subsections) and a simple realizability relation (that will be denoted by \Vdash^\bullet) in the sense of the \Vdash^0

relation of Subsection 3.7. In both cases, it will be implicit that the interpretation of constructors remain the same as the one defined in Section 3.

Equational implication — To define the formula translation, we follow [22] and add a new type constructor to $\mathbf{MAL}\omega$. If A is a formula of kind o^+ (resp. o^-), and if T and U are two type constructors of the same kind, then we have a new type constructor of kind o^+ (resp. o^-) denoted $\langle T = U \rangle A$. Its informal meaning is “ $T \cong U$ implies A ”. Even if we can define the forcing translation without it, it will simplify the proof of the connection lemma. We will not add the corresponding typing rules, because we will deal directly with (non quantitative) realizability. For any valuation ρ , the realizability interpretation of section 3 is extended to the new formula $\langle T = U \rangle A$ as follows:

$$|\langle T = U \rangle A|_\rho = \begin{cases} |A|_\rho & \text{if } T \cong U \\ \emptyset^\perp & \text{else} \end{cases}$$

5.2. Linear forcing structures

To compose realizability and forcing, we formalize a forcing interpretation *inside* $\mathbf{MAL}\omega$. We mostly follow Krivine’s formulation of forcing [14, 22]. We begin by giving a *linear* version of Krivine’s forcing structure.

Definition 68 (Linear forcing structure). A *linear forcing structure* is given by the following components:

- κ , the kind of *conditions*.
- $\mathcal{C}[\cdot] : \kappa \rightarrow o^\circ$, with $\circ \in \{+, -\}$ is a positive or negative predicate.
- $\mathbf{0} : \kappa$ is a distinguished condition.
- $+$: $\kappa \rightarrow \kappa \rightarrow \kappa$ is a binary operation on conditions, such that for every $p, q, r : \kappa$, the following conversions hold in $\mathbf{MAL}\omega$:

$$\begin{aligned} p + (q + r) &\cong (p + q) + r \\ p + q &\cong q + p \\ \mathbf{0} + p &\cong p \end{aligned}$$

Example 69. A simple example of linear forcing structure is the integer forcing structure, defined as follows:

- The kind of conditions is the kind ι of integers.
- The predicate is $\lambda x. \top : \iota \rightarrow o^-$.
- $+$ is the usual addition on integers defined using rec_ι .
- $\mathbf{0}$ is the constructor $\mathbf{0} : \iota$.

Then it is clear that for every integers $p, q, r : \iota$, the requested conversions hold in $\mathbf{MAL}\omega$.

Remark 70. What we note $+$ and $\mathbf{0}$ is written $.$ and $\mathbf{1}$ in [14, 22]. We choose an additive notation instead of a multiplicative one, because we think it better fits the quantitative intuition of multiplicative linear logic. It has also the advantage of being closer to the symbols used in the definition of quantitative monoids.

The linear forcing structure is a sharp simplification of the notion of forcing structure as defined in [14], in particular because we ask κ to be a monoid with respect to \cong (that is, at the computational level instead of the provability level). This is however sufficient for our purpose.

Remark 71. Informally, $\mathcal{C}[p+q]^\perp$ represents a notion of orthogonality between p and q , and plays the same role in forcing as the pole \perp in realizability. Observe that a linear forcing structure (modulo \cong) is a multiplicative phase space [7], by choosing $\mathcal{C}[\cdot]^\perp$ as the pole.

5.3. Formula translation

We assume having fixed a linear forcing structure on the kind κ . We now formalize inside $\mathbf{MAL}\omega$ a forcing interpretation of \mathbf{MAL} . Following [22] methodology, we associate to each \mathbf{MAL} formula A a $\mathbf{MAL}\omega$ formula $p \Vdash_f A$ (which is read " p forces A "). Because in the rest of this paper all the quantifications are made on κ , we omit to indicate the kinds on the quantifiers and on the variables of kind κ .

Definition 72. Let $Z : \kappa \rightarrow o^\circ$ (for $\circ \in \{+, -\}$). Then the *forcing orthogonal* of Z is defined as a $\mathbf{MAL}\omega$ constructor of kind $\kappa \rightarrow o^-$:

$$\overline{Z} \equiv \lambda r. \forall r'. Z(r') \multimap \mathcal{C}[r + r']$$

Remarks 73.

1. The definition of forcing orthogonal is dependent of the choice of the linear forcing structure, since it depends of the kind κ and the choice of the predicate $\mathcal{C}[\cdot]$.
2. Notice that the polarity of the predicate $\overline{Z} : \kappa \rightarrow o^-$ does not depend of the polarity of the predicate Z : if Z is a positive or negative predicate on κ , then \overline{Z} is a negative predicate on κ . This is a consequence of our choice of a negative encoding of the \multimap connective.

We now define the forcing translation. We suppose that we associate to every \mathbf{MAL} variable X a $\mathbf{MAL}\omega$ variable $X^{\kappa \rightarrow o^+}$ of kind $\kappa \rightarrow o^+$. If $A : o^\circ$ (with $\circ \in \{+, -\}$) is a \mathbf{MAL} formula, we define a $\mathbf{MAL}\omega$ constructor $A^* : \kappa \rightarrow o^\circ$

inductively as follows:

$$\begin{aligned}
X^* &\equiv X^{\kappa \rightarrow o^+} \\
(X^\perp)^* &\equiv \overline{X^{\kappa \rightarrow o^+}} \\
(A \otimes B)^* &\equiv \overline{\lambda r. \exists p_1. \exists p_2. \langle r = p_1 + p_2 \rangle (A^*(p_1) \otimes B^*(p_2))} \\
(A \wp B)^* &\equiv \overline{\lambda r. \exists p_1. \exists p_2. \langle r = p_1 + p_2 \rangle ((A^\perp)^*(p_1) \otimes B^{\perp*}(p_2))} \\
(\downarrow A)^* &\equiv \overline{\lambda r. \downarrow A^*(r)} \\
(\uparrow A)^* &\equiv \overline{\lambda r. \downarrow (A^\perp)^*(r)}
\end{aligned}$$

Finally, if A is a **MAL** formula and $p : \kappa$, we define $p \Vdash_f A$ as a **MAL** ω constructor of kind o^- as follows:

$$\begin{aligned}
p \Vdash_f P &\equiv \overline{\overline{P^*}}(p) \\
p \Vdash_f N &\equiv N^*(p)
\end{aligned}$$

Remarks 74.

1. Informally, A^* and $p \Vdash_f A$ have respectively the same role as the sets $\|A\|$ and $|A|$ defined in Subsection 3.4.
2. The predicate A^* has the same polarity as A . However, the formula $p \Vdash_f A$ is always negative, even if A is positive.

In the formalization of the forcing orthogonal, we use a negative encoding of the \multimap connective: this is an adaptation of the negative forcing translation defined in [22, 14]. We could have defined a positive forcing translation, but what really matters is the polarity of $\mathcal{C}[\cdot]$. Whereas in [22, 14] $\mathcal{C}[\cdot]$ is always negative, we allow it to be positive.

Property 75.

1. For every negative formula N , we have $N^*(p) \cong \overline{N^{\perp*}}(p)$.
2. For every positive formula P and every $p : \kappa$, we have

$$p \Vdash_f P \cong \overline{\lambda r. (r \Vdash_f P^\perp)}(p)$$

5.4. The countdown machine

We now describe a new abstract machine. Because of the mechanism it implements, we call it the *countdown machine*. Although this machine is based on the term syntax of \mathbf{L}_{foc} , it has completely different reduction rules and hence is *not* just another extension of \mathbf{L}_{foc} . We now suppose that the set \mathcal{K}_+ contains new instructions constant \overline{n} for each $n \in \mathbb{N}$. Hence term syntax is augmented with *primitive integers*. We denote by \mathbb{N}^\bullet the set $\{ \overline{n} \mid n \in \mathbb{N} \}$.

Definition 76. To describe the evaluation in this machine, we need to consider two new kinds of commands:

1. *Negative commands* are of the form $\langle t^\ominus \mid u^+ \rangle$ where t is a positive or negative term whereas u^+ is a positive term. The set of negative commands is denoted by \mathcal{C}^\ominus .
2. *Forcing commands* are negative commands of the form $\langle t^\ominus \mid (u, K) \rangle$, where:
 - $K \in \mathcal{T}$ is a term (either positive or negative).
 - $t, u \in \mathcal{T}$ are terms of opposite polarities.

Such a forcing command will be sometimes noted $\langle t^\ominus \mid u \rangle \{K\}$.

If $K \in \mathcal{T}$, $t^+ \in \mathcal{T}_+^0$ and $u^- \in \mathcal{T}_-^0$ are respectively a positive and a negative term, we can build from the command $\langle t^+ \mid u^- \rangle$ a forcing command noted $\langle t^+ \mid u^- \rangle^\bullet \{K\} = \langle u^{-\ominus} \mid (t^+, K) \rangle$.

Remark 77. The notation t^\ominus is just a marker on t to indicate that it is now considered as negative. This is due to the formula translation. Indeed, all the translated formulas are negative, so even if a term t is positive, its *image* in the machine is negative and can be executed in front of a positive term.

The reduction relation in this machine is denoted by \rightarrow_\bullet , and is defined between forcing commands by the following rules:

$$\begin{array}{ll}
\langle t^{-\ominus} \mid (\mu\alpha.c) \rangle \{\overline{n}\} & \rightarrow_\bullet \quad (c[t^-/\alpha])^\bullet \{\overline{n}\} \\
\langle (\mu x.c)^\ominus \mid V^+ \rangle \{\overline{n}\} & \rightarrow_\bullet \quad (c[V^+/x])^\bullet \{\overline{n}\} \\
\langle (\mu(\kappa, \kappa').c)^\ominus \mid (V_1, V_2) \rangle \{\overline{n+1}\} & \rightarrow_\bullet \quad (c[V_1/\kappa, V_2/\kappa'])^\bullet \{\overline{n}\} \\
\langle (\mu\{\kappa\}.c)^\ominus \mid \{V\} \rangle \{\overline{n+1}\} & \rightarrow_\bullet \quad (c[V/\kappa])^\bullet \{\overline{n}\} \\
\langle (\mu(\kappa, \kappa').c)^\ominus \mid (V_1, V_2) \rangle \{\overline{0}\} & \Uparrow^\bullet \\
\langle (\mu\{\kappa\}.c)^\ominus \mid \{V\} \rangle \{\overline{0}\} & \Uparrow^\bullet
\end{array}$$

Remark 78. These rules indeed implement a kind of countdown: each step makes the counter decrease, and if the counter equals 0 then any step makes the machine diverge.

In the same spirit of the identification of $\langle t \mid u \rangle$ and $\langle u \mid t \rangle$, we quotient the set of forcing commands by the following α -equivalence:

$$\langle u^\ominus \mid t \rangle \{K\} \equiv \langle t^\ominus \mid u \rangle \{K\}$$

It must be remarked that if c is a command and $K \notin \mathbb{N}^\bullet$, then $c^\bullet \{K\}$ never reduces for \rightarrow_\bullet .

Remark 79. In contrast with [22] we don't define any program transformation to justify the reduction rules of the machine. The justification of the introduction of the machine will be given *a posteriori*, by a specific linear forcing structure.

5.5. A countdown machine-based realizability model

We now describe the realizability interpretation that is induced by the countdown machine. It is a simple realizability, in the sense that it relates a term t and a **MAL** ω constructor T . It will be used to state the **connection lemma**. This realizability relation is *parametrized* by a set \perp^\bullet of forcing negative commands closed under anti \rightarrow_\bullet -evaluation:

$$\forall c, c' \in \mathcal{C}^\ominus, \text{ if } c \rightarrow_\bullet c' \text{ and } c' \in \perp^\bullet \text{ then } c \in \perp^\bullet$$

Remark 80. Because of the α -equivalence on forcing commands, the following equivalence holds: $\langle t^\ominus \mid (u, K) \rangle \in \perp^\bullet \Leftrightarrow \langle u^\ominus \mid (t, K) \rangle \in \perp^\bullet$.

We now suppose that such a set \perp^\bullet is fixed. Based on this set, we want to define a simple realizability interpretation. Since \perp^\bullet is not a set of commands, it is impossible to reuse immediately the definitions of Section 3. But we can define a new a set \perp^\bullet_0 of commands (in the usual sense) out of it:

$$\perp^\bullet_0 \equiv \{ \langle t^+ \mid u^- \rangle \mid \langle u^{-\ominus} \mid t^+ \rangle \in \perp^\bullet \}$$

Now, suppose we have fixed a propositional domain $\mathcal{D}^{\oplus\bullet}$ and a total valuation ρ^\bullet . Although \perp^\bullet_0 is not closed under anti-evaluation for \rightarrow_0 , and hence is not saturated, we can still consider the interpretation it induces. We then obtain an interpretation of **MAL** ω constructor. For each constructor T , this interpretation is denoted $\|T\|_{\rho^\bullet}^\bullet$, and the associated realizability relation is denoted \Vdash^\bullet . In particular we have:

$$t \Vdash^\bullet N \iff \text{for every } u \Vdash^\bullet N^\perp \text{ we have } \langle t^\ominus \mid u \rangle \in \perp^\bullet$$

Remarks 81.

1. If the interpretation of a **MAL** ω constructor can still be defined, neither the adequacy theorem with respect to **MAL** ω nor the properties of Subsection 3.5 are valid.
2. The adequacy result, as stated in Subsection 3.7, does not hold. However, we can and will prove a different adequacy result stated using forcing.
3. Finally, it has to be noted that if P is a positive formula and $t \in \mathcal{T}$ is a term, then $t \in \|P\|_{\rho^\bullet}^\bullet$ implies that t is positive. However, if N is a negative formula, $t \in \|N\|_{\rho^\bullet}^\bullet$ does not imply that t is negative, as we will see in Subsection 5.7.

While the identification $\langle t \mid u \rangle = \langle u \mid t \rangle$ is reminiscent of the involutivity of the linear negation, the new α -equivalence corresponds to an identification between a term of the form $\overline{X} : \kappa \rightarrow o^-$ and its forcing biorthogonal $\overline{\overline{\overline{X}}}$. Indeed,

Property 82. Suppose $T : \kappa \rightarrow o^+$, then if $p : \kappa$, the following holds:

1. $t \Vdash^\bullet T(p)$ implies $t \Vdash^\bullet \overline{\overline{\overline{T}}}(p)$

2. $t \Vdash^\bullet \overline{\overline{T}}(p)$ implies $t \Vdash^\bullet \overline{T}(p)$

PROOF.

1. Suppose $t \Vdash^\bullet T(p)$. Then take $K \in \|\mathcal{C}[p+r]\|^\bullet$ for some $r : \kappa$ and $u \in \|\overline{T}(r)\|_{\rho^\bullet}^\bullet$. Then because $\overline{T}(r) = \forall r'. T(r')^\perp \wp \mathcal{C}[r+r']^\perp$, we have $\langle u^\ominus \mid (t, K) \rangle \in \perp^\bullet$. But, by α -equivalence, we know that $\langle t^\ominus \mid (u, K) \rangle \in \perp^\bullet$. Hence, $t \Vdash^\bullet \overline{\overline{T}}(p)$.
2. If $t \Vdash^\bullet \overline{\overline{T}}(p)$, $K \in \|\mathcal{C}[p+r]\|^\bullet$, $u \in \|T(r)\|_{\rho^\bullet}^\bullet$, then by the previous point, $u \in \|\overline{\overline{T}}(r)\|_{\rho^\bullet}^\bullet$, so $\langle t^\ominus \mid (u, K) \rangle \in \perp^\bullet$ which concludes.

Hence, as a corollary we immediately obtain that the two following rules are adequate with respect to the $\|\cdot\|^\bullet$ interpretation:

$$\frac{\vdash t : T(p) \mid \Gamma}{\vdash t : \overline{\overline{T}}(p) \mid \Gamma} \quad \frac{\vdash t : \overline{\overline{T}}(p) \mid \Gamma}{\vdash t : \overline{T}(p) \mid \Gamma}$$

5.6. A quantitative linear forcing structure

We now describe a particular linear forcing structure. The relation obtained by composition of the forcing translation induced by this structure and the realizability based on the countdown machine of Subsection 5.5 will be shown in next subsection to coincide with a quantitative realizability relation. The structure considered is $(\iota, \mathcal{C}[\cdot], +, \mathbf{0})$ where:

- The kind of conditions is ι , the kind of natural numbers.
- $\mathcal{C}[\cdot]$ is a new predicate of kind $\iota \rightarrow o^+$.
- $+$ is the usual addition on natural numbers, defined using rec_ι :

$$\cdot + \cdot = \lambda p^\iota \lambda q^\iota . rec_\iota \, p \, \mathbf{s} \, q$$

- $\mathbf{0}$ is the corresponding individual.

Property 83. $(\iota, \mathcal{C}[\cdot], +, \mathbf{0})$ is a linear forcing structure.

PROOF. Associativity, commutativity and neutrality of $\mathbf{0}$ with respect to $+$ are easily checked. As an example, we show the neutrality of $\mathbf{0}$. We first notice that using the rules of Figure 2, $\mathbf{0} + q \cong rec_\iota \, \mathbf{0} \, \mathbf{s} \, q$. But we also have $rec_\iota \, \mathbf{0} \, \mathbf{s} \, q \cong q$. Hence, by transitivity of \cong we have $\mathbf{0} + q \cong q$.

As $\mathcal{C}[\cdot]$ is a new (positive) predicate of kind $\iota \rightarrow o^+$, we need to say what its realizability interpretation is. For each valuation ρ^\bullet , we pose:

$$\|\mathcal{C}[\cdot]\|_{\rho^\bullet}^\bullet = p \in \mathbb{N} \mapsto \{ \overline{n} \mid p \leq n \wedge n \in \mathbb{N} \}$$

Since this function does not depend of the valuation ρ^\bullet , we will not write the ρ^\bullet and note it $\|\mathcal{C}[\cdot]\|^\bullet$.

Remarks 84.

1. We will often switch between concrete elements of ι and elements of \mathbb{N} . As already mentioned in Section 1, we will denote \mathbf{n} the element of kind ι corresponding to the integer $n \in \mathbb{N}$, which avoids confusion.
2. For the interpretation to make sense, we need $\mathcal{D}^{\oplus \bullet}$ to contain all the sets $\|\mathcal{C}[p]\|^\bullet$ with $p : \iota$. From now on, we will only consider such $\mathcal{D}^{\oplus \bullet}$.
3. This linear forcing structure can be compared to the quantitative monoid of integers described in Example 15. Indeed, we will see in Subsection 5.8 that they play the same role.
4. Similarly, many quantitative monoids can be turned into linear forcing structures. For example, suppose that we have extended the language of kinds with a product kind $\kappa \times \kappa'$ and added the pair (T, U) and projections π_i constructors (it is not difficult to see how to extend the realizability model to such a framework). Then, the soft monoid can be described as a linear forcing structure defined as follows:
 - The kind is $\kappa = \iota \times (\iota \rightarrow \iota)$
 - The maximum max of two elements of ι is easily defined using rec_ι , and the addition $+_s$ of the soft monoid can then be defined using max :

$$+_s = \lambda x^\kappa. \lambda y^\kappa. (max(\pi_1 x^\kappa, \pi_1 y^\kappa), \lambda z^\iota. max(\pi_2 x^\kappa(z), \pi_2 y^\kappa(z)))$$

- $\|\mathcal{C}[p]\|^\bullet = \{ \overline{n} \mid \|p\| \leq n \wedge n \in \mathbb{N} \}$ where

$$\|\cdot\| = \lambda x^\kappa. (\pi_2 x^\kappa)(\pi_1 x^\kappa)$$

5.7. A connection theorem

In this subsection, the connection between quantitative realizability and forcing is set out in the form of a **connection theorem**, which states that the composition of the forcing relation induced by the quantitative linear structure of Subsection 5.6 and countdown machine based realizability of Subsection 5.5 yields a quantitative realizability model of **MAL**. We then use this result together with Theorem 42 to obtain an adequacy result for linear forcing.

We suppose having fixed a set \perp^\bullet which is closed under anti \rightarrow_\bullet -evaluation and the associated set of commands \perp^\bullet_0 . We also suppose having a propositional domain $\mathcal{D}^{\oplus \bullet}$ and a total valuation ρ^\bullet . We suppose having fixed a set \perp^\bullet which is closed under anti \rightarrow_\bullet -evaluation and the associated set of commands \perp^\bullet_0 . We also suppose having a propositional domain $\mathcal{D}^{\oplus \bullet}$ and a total valuation ρ^\bullet .

Definition 85. We define the following objects:

- $\perp^\circ = \{ (c, p) \mid \forall K \in \|\mathcal{C}[\mathbf{p}]\|^\bullet, c^\ominus \{K\} \in \perp^\bullet \}$
- $\rho^\circ(X) = \{ (t, p) \mid t \in \rho^\bullet(X^{\iota \rightarrow o^+})(p) \}$

- $(\mathcal{D}^\oplus)^\circ = \{ X \mid \exists Y \in (\mathcal{D}^{\oplus \bullet})^\mathbb{N} \text{ such that } (t, p) \in X \Leftrightarrow t \in Y(\mathbf{p}) \}$

Hence, ρ° is a valuation, which is not total but defined on all **MAL** variables. Since we are only interested in **MAL** variables, but need a total one to reuse the interpretation defined in Section 3, we will in fact consider a valuation whose restriction on **MAL** variables is ρ° and identify it with ρ° .

Because \perp^\bullet is closed under anti- \rightarrow_\bullet evaluation, \perp° yields a saturated quantitative pole, as witnessed by the following property.

Property 86. $(\mathbb{N}, \perp^\circ, 1)$ is a saturated quantitative pole.

PROOF. Suppose $(c', p) \in \perp^\circ$ and $c \rightarrow_0 c'$. Then we want to prove that for any $K \in \|\mathcal{C}[\mathbf{p} + \mathbf{1}]\|^\bullet$, $c\{K\} \in \perp^\bullet$. But $K = \bar{n}$ and $p + 1 \leq n$ so $n = n' + 1$ and $K = \bar{n'} + 1$. So $c^\bullet\{\bar{n'} + 1\} \rightarrow_\bullet c'^\bullet\{\bar{n'}\}$ with $\bar{n'} \in \|\mathcal{C}[\mathbf{p}]\|^\bullet$, so $c'^\bullet\{\bar{n'}\} \in \perp^\bullet$ and by anti-reduction property, we obtain the conclusion.

Property 87. $(\mathcal{D}^\oplus)^\circ$ is a positive propositional domain.

PROOF. It is clear since $\mathcal{D}^{\oplus \bullet}$ is itself a positive propositional domain.

Since \perp° is a quantitative pole, $\mathcal{D}^{\oplus \circ}$ is a propositional domain and ρ° is a total valuation, we obtain a quantitative realizability interpretation of **MAL** ω , as defined in Section 3. We denote this new interpretation $\|\cdot\|_{\rho^\circ}^\circ$, and the associated realizability interpretation \Vdash° .

Remark 88. We now have three different interpretations of **MAL** formulas A :

- The quantitative interpretation $\|A\|_{\rho^\circ}^\circ$, which is a set of bounded terms.
- The non-quantitative interpretation $\|A\|_{\rho^\bullet}^\bullet$, which is based on the count-down machine and is a set of terms (in fact, bounded terms where the bound is an element of the trivial monoid $\{0\}$).
- The forcing interpretation $A^*(p)$, which is a **MAL** ω formula.

All these interpretations are related through the following **connection lemma**:

Lemma 89. For every **MAL** formula C , every positive propositional domain $\mathcal{D}^{\oplus \bullet}$ and every total valuation ρ^\bullet , we have

$$t \in \|C^*(\mathbf{p})\|_{\rho^\bullet}^\bullet \iff (t, p) \in \|C\|_{\rho^\circ}^\circ$$

PROOF. The proof is carried out by induction on the formula C . For each case we prove directly the equivalence.

- If $C = X$,

$$\begin{aligned} t \in \|X^*(\mathbf{p})\|_{\rho^\bullet}^\bullet &\iff t \in \|X^{\iota \rightarrow o^+}\|_{\rho^\bullet}^\bullet(p) \\ &\iff t \in \rho^\bullet(X^{\iota \rightarrow o^+})(p) \\ &\iff (t, p) \in \rho^\circ(X) \\ &\iff (t, p) \in \|X\|_{\rho^\circ}^\circ \end{aligned}$$

- If $C = X^\perp$,

$$\begin{aligned}
& t \in \|(X^\perp)^*(\mathbf{p})\|_{\rho^\bullet}^\bullet \\
\iff & t \in \|\forall r. X^*(r) \multimap \mathcal{C}[\mathbf{p} + r]^\perp\|_{\rho^\bullet}^\bullet \\
\iff & \forall r \in \mathbb{N}, \forall u \in \rho^\bullet(X^{\iota \rightarrow o^+})(r), \forall K \in \|\mathcal{C}[\mathbf{p} + r]\|^\bullet, \langle t^\ominus \mid (u, K) \rangle \in \mathbb{L}^\bullet \\
\iff & \forall r \in \mathbb{N}, \forall (u, r) \in \rho^\circ(X), \forall K \in \|\mathcal{C}[\mathbf{p} + r]\|^\bullet, \langle t^\ominus \mid (u, K) \rangle \in \mathbb{L}^\bullet \\
\iff & (t, p) \in \rho^\circ(X)^\perp \\
\iff & (t, p) \in \|X^\perp\|_{\rho^\circ}^\circ
\end{aligned}$$

In the remaining cases, we do not write the valuations ρ° and ρ^\bullet in the interpretations, since they play no role here.

- If $C = \downarrow A$,

$$\begin{aligned}
t \in \|\downarrow A^*(\mathbf{p})\|_{\rho^\bullet}^\bullet & \iff t = \{t'\} \text{ and } t' \in \|A^*(\mathbf{p})\|_{\rho^\bullet}^\bullet \\
& \iff t = \{t'\} \text{ and } (t', p) \in \|A\|_{\rho^\circ}^\circ \\
& \iff (t, p) \in \|\downarrow A\|_{\rho^\circ}^\circ
\end{aligned}$$

- If $C = \uparrow A$,

$$\begin{aligned}
& t \in \|(\uparrow A)^*(\mathbf{p})\|^\bullet \\
\iff & t \in \|\forall x^\iota. (\downarrow A^\perp)^*(x) \multimap \mathcal{C}[\mathbf{p} + x]^\perp\|^\bullet \\
\iff & \forall q \in \mathbb{N}, t \in \|\downarrow (A^\perp)^*(\mathbf{q}) \otimes \mathcal{C}[\mathbf{p} + \mathbf{q}]\|_{\rho^\bullet}^{\bullet\perp} \\
\iff & \forall q \in \mathbb{N}, \forall K \in \|\mathcal{C}[\mathbf{p} + \mathbf{q}]\|^\bullet, \forall u \in \|(A^\perp)^*\|^\bullet(q), \langle t^\ominus \mid (\{u\}, K) \rangle \in \mathbb{L}^\bullet \\
\iff & \forall (u, q) \in \|A^\perp\|_{\rho^\circ}^\circ, \forall K \in \|\mathcal{C}[\mathbf{p} + \mathbf{q}]\|^\bullet, \langle t^\ominus \mid (\{u\}, K) \rangle \in \mathbb{L}^\bullet \\
\iff & \forall (u, q) \in \|A^\perp\|_{\rho^\circ}^\circ, (t, p) \perp^\circ(\{u\}, q) \\
\iff & (t, p) \in \|\uparrow A\|_{\rho^\circ}^\circ = \|\downarrow A^\perp\|_{\rho^\circ}^{\circ\perp}
\end{aligned}$$

- If $C = A \otimes B$,

$$\begin{aligned}
& t \in \|(A \otimes B)^*(\mathbf{p})\|^\bullet \\
\iff & \exists p_1, p_2 \in \mathbb{N}, p = p_1 + p_2 \wedge t \in \|A^*(\mathbf{p}_1)\|^\bullet \otimes \|B^*(\mathbf{p}_2)\|^\bullet \\
\iff & \exists p_1, p_2 \in \mathbb{N}, p = (p_1, p_2) \wedge t = (t_1, t_2) \wedge t_1 \in \|A^*(\mathbf{p}_1)\|^\bullet \wedge t_2 \in \|B^*(\mathbf{p}_2)\|^\bullet \\
\iff & \exists p_1, p_2 \in \mathbb{N}, p = (p_1, p_2) \wedge t = (t_1, t_2) \wedge (t_1, p_1) \in \|A\|_{\rho^\circ}^\circ \wedge (t_2, p_2) \in \|B\|_{\rho^\circ}^\circ \\
\iff & (t, p) \in \|A \otimes B\|_{\rho^\circ}^\circ
\end{aligned}$$

- If $C = A \wp B$,

$$\begin{aligned}
& t \in \|(A \wp B)^*(\mathbf{p})\|^\bullet \\
\iff & \forall u_1 \in \|(A^\perp)^*(\mathbf{p}_1)\|^\bullet, \forall u_2 \in \|(B^\perp)^*(\mathbf{p}_2)\|^\bullet, \forall K \in \|\mathcal{C}[\mathbf{p} + \mathbf{p}_1 + \mathbf{p}_2]\|^\bullet, \\
& \quad \langle t^\ominus \mid ((u_1, u_2), K) \rangle \in \mathbb{L}^\bullet \\
\iff & \forall K \in \|\mathcal{C}[\mathbf{p} + \mathbf{p}_1 + \mathbf{p}_2]\|^\bullet, \forall (u_1, p_1) \in \|A^\perp\|_{\rho^\circ}^\circ, \forall (u_2, p_2) \in \|B^\perp\|_{\rho^\circ}^\circ, \\
& \quad \langle t^\ominus \mid ((u_1, u_2), K) \rangle \in \mathbb{L}^\bullet \\
\iff & \forall (u_1, p_1) \in \|A^\perp\|_{\rho^\circ}^\circ, \forall (u_2, p_2) \in \|B^\perp\|_{\rho^\circ}^\circ, (t, p) \perp((u_1, u_2), p_1 + p_2) \\
\iff & (t, p) \in \|A \wp B\|_{\rho^\circ}^\circ
\end{aligned}$$

As a corollary of this lemma and of Property 82, we obtain the following connection theorem.

Theorem 90 (Connection theorem). *For every MAL formula C and for every $t \in \mathcal{T}$ and $p \in \mathbb{N}$, we have*

$$t \Vdash^\bullet (\mathbf{p} \Vdash_f C) \iff (t, p) \Vdash^\circ C$$

PROOF. We use the previous theorem. Let's distinguish two cases, depending of the polarity of C .

1. Suppose $C = P$ is positive. Then:

$$\begin{aligned} t \Vdash^\bullet (\mathbf{p} \Vdash_f P) &\iff t \in \|\overline{P^*}(\mathbf{p})\|^\bullet \\ &\iff t \in \|\forall r. (\overline{P^*}(r) \multimap \mathcal{C}[\mathbf{p} + r]^\perp)\|^\bullet \\ &\iff \forall r \in \mathbb{N}. \forall u \in \|\overline{P^*}(\mathbf{r})\|^\bullet, \forall K \in \|\mathcal{C}[\mathbf{p} + \mathbf{r}]\|^\bullet, \langle t^\ominus \mid (u, K) \rangle \in \mathbb{L}^\bullet \\ &\iff \forall r \in \mathbb{N}. \forall u \in \|\overline{P^*}(\mathbf{r})\|^\bullet, \forall K \in \|\mathcal{C}[\mathbf{p} + \mathbf{r}]\|^\bullet, \langle t^\ominus \mid (u, K) \rangle \in \mathbb{L}^\bullet \end{aligned}$$

Since P is positive, we have $(P^\perp)^* = \overline{P^*}$. Hence:

$$\begin{aligned} &\forall r \in \mathbb{N}. \forall u \in \|\overline{P^*}(\mathbf{r})\|^\bullet, \forall K \in \|\mathcal{C}[\mathbf{p} + \mathbf{r}]\|^\bullet, \langle t^\ominus \mid (u, K) \rangle \in \mathbb{L}^\bullet \\ \Leftrightarrow &\forall (u, r) \in \|\overline{P^\perp}\|^\circ, \forall K \in \|\mathcal{C}[\mathbf{p} + \mathbf{r}]\|^\bullet, \langle t^\ominus \mid (u, K) \rangle \in \mathbb{L}^\bullet \end{aligned}$$

But if $\langle t^\ominus \mid (u, K) \rangle \in \mathbb{L}^\bullet$ and u is a negative term, then it implies that t is a positive term. Indeed \mathbb{L}^\bullet is a set of forcing negative commands. Hence,

$$\begin{aligned} &\forall (u, r) \in \|\overline{P^\perp}\|^\circ, \forall K \in \|\mathcal{C}[\mathbf{p} + \mathbf{r}]\|^\bullet, \langle t^\ominus \mid (u, K) \rangle \in \mathbb{L}^\bullet \\ \Leftrightarrow &(t, p) \in \|\overline{P^\perp}\|^\circ \\ \Leftrightarrow &(t, p) \Vdash^\circ P \end{aligned}$$

2. Suppose $C = N$ is negative. Then $\overline{N^*} = \overline{(N^\perp)^*}$. We have

$$\begin{aligned} t \Vdash^\bullet (\mathbf{p} \Vdash_f N) &\iff t \in \|\overline{N^*}(\mathbf{p})\|^\bullet \\ &\iff t \in \|\overline{(N^\perp)^*}(\mathbf{p})\|^\bullet \end{aligned}$$

By Property 75, we have

$$t \in \|\overline{(N^\perp)^*}(\mathbf{p})\|^\bullet \iff t \in \|\overline{(N^\perp)^*}(\mathbf{p})\|^\bullet$$

By Property 82, we also have the following equivalence:

$$t \in \|\overline{(N^\perp)^*}(\mathbf{p})\|^\bullet \iff t \in \|N^*(\mathbf{p})\|^\bullet$$

Finally, by Lemma 89, we obtain

$$t \in \|N^*(\mathbf{p})\|^\bullet \iff (t, p) \in \|N\|^\circ$$

Since, N is negative, $\|N\|^\circ = \|N\|^{\circ\perp\perp}$ and hence we obtain

$$t \Vdash^\bullet (\mathbf{p} \Vdash_f N) \iff (t, p) \Vdash^\circ N$$

Remarks 91.

1. This theorem shows that positive terms t can realize (in the sense of the $\|\cdot\|^\bullet$ interpretation) a negative formula. Indeed, if P is positive and $p \in \mathbb{N}$ then $p \Vdash_f P$ is always a negative formula, realized by a positive term.
2. This last theorem show the connection between quantitative linear forcing and quantitative realizability. It says considering this specific linear forcing structure *inside* the countdown realizability model (for any choice of a pole) is equivalent to a certain quantitative realizability on **MAL**.
3. If we have shown that composing the linear forcing and countdown realizability induces a quantitative realizability relation, the converse is not true. Indeed, a quantitative realizability relation is not a priori equivalent to the composition of a certain forcing and a countdown realizability.

By Property 86, we know that \perp° is a saturated quantitative pole. Hence, Theorem 42 is valid. Together with the connection theorem, it can be used to obtain an adequacy theorem for linear forcing with respect to **MAL**, *inside* the realizability model.

Theorem 92. *Suppose A, B_1, \dots, B_n are **MAL** formulas. Suppose π is a proof of $(\vdash t : A \mid \kappa_1 : B_1, \dots, \kappa_n : B_n)$. Let $u_1, \dots, u_n \in \mathcal{T}$ and $q_1, \dots, q_n \in \mathbb{N}$ such that for any $i \in \llbracket 1, n \rrbracket$ we have $u_i \Vdash^\bullet (q_i \Vdash_f B_i)$. Then if we pose $p = \mathbf{M}[\pi]$, we have*

$$t[u_1/\kappa_1, \dots, u_n/\kappa_n] \Vdash^\bullet (\mathbf{p} + \mathbf{q}_1 + \dots + \mathbf{q}_n \Vdash_f A)$$

This result justifies *a posteriori* the introduction of the countdown machine.

5.8. Forcing and reducibility candidates

We have seen that it is possible to obtain certain instances of the quantitative realizability by composing forcing and countdown machine-based realizability. We now show an example of such an instance: the quantitative reducibility candidates of Subsection 3.8 restricted to **MAL**, arise from the composition of our quantitative linear forcing of Subsection 5.6 and non quantitative reducibility candidates, adapted to the countdown machine.

We choose a set \perp_{Time} of bounded commands, a positive propositional domain \mathcal{D}^\oplus and a total valuation ρ such that:

$$\begin{aligned} \perp_{Time} &= \{ (c, n) \mid c \text{ normalizes for } \rightarrow_0^* \text{ using at most } n \beta\text{-steps} \} \\ \rho(X^{o^+}) &= \{ (\boxplus_+, p) \mid p \in \mathbb{N} \} \end{aligned}$$

Those correspond to a particular case of quantitative reducibility candidates of Subsection 3.8. Indeed, in the case of the quantitative monoid of integers, $\rho(X^{o^+})$ is a quantitative reducibility candidate. We denote the corresponding realizability relation \Vdash_{Time} .

On another hand, we can define the (non-quantitative) reducibility candidates model corresponding to the countdown machine. It is an instance of the countdown machine based realizability. We choose the following \Vdash^\bullet and the valuation ρ^\bullet :

$$\begin{aligned}\Vdash^\bullet &= \{ c \in \mathcal{C}^\ominus \mid c \text{ normalizes for } \rightarrow_\bullet \} \\ \rho^\bullet(X^{\iota \rightarrow o^+})(p) &= \{\mathbf{x}_+\}\end{aligned}$$

It is clear that these definitions correspond to those of Subsection 3.8 where the quantitative part has been erased.

Property 93.

1. We have $\Vdash_{Time} = \Vdash^\circ$.
2. For every **MAL** atom X^{o^+} , we have $\rho^\circ(X^{o^+}) = \rho(X^{o^+})$.

PROOF.

1. If t_+ and u_- are two terms, then $\langle t_+ \mid u_- \rangle$ normalizes in a number of \rightarrow_β -steps at most p if and only if for any $n \in \mathbb{N}$ greater than p , $\langle (u_-)^\ominus \mid (\bar{n}, t^+) \rangle$ normalizes for \rightarrow_\bullet .
2. This is by definition.

As an immediate corollary of Property 93 and Lemma 89, we obtain the following **decomposition theorem**:

Theorem 94. *If A is a **MAL** formula, then*

$$(t, n) \Vdash_{\mathbf{Time}} A \iff t \Vdash^\bullet (\mathbf{n} \Vdash_f A)$$

5.9. Remarks

Let's finish by a few remarks about this forcing decomposition, and the choices we have made.

Exponentials and quantifiers — Although we have only treated **MAL** in this section, we could extend these results to a system with exponentials, like **SAL** ω . To do so, it suffices to give an interpretation of $!$ and $?$. The definition would be:

$$\begin{aligned}(!A)^* &= \lambda x^\kappa. \forall q. \langle x = !q \rangle (! (A^*(q))) \\ (?A)^* &= \lambda x^\kappa. \overline{\lambda r. \forall q. \langle r = !q \rangle (! ((A^\perp)^*(q)))}(x)\end{aligned}$$

where $!$ is a term of kind $\kappa \rightarrow \kappa$. Of course, properties of $\mathcal{C}[\cdot]$ with respect to $!$ would be needed. Concerning the quantifiers, if one wants to interpret them, it

suffices to follow the construction described in [22].

The choice of the machine — In the countdown machine, when the counter reaches 0, any β -step makes it diverge. This is a necessary choice we have made in order to obtain Property 93. We could have chosen any other behavior without losing Lemma 89 and Theorem 92. In particular, we could define a machine that executes programs for a certain number of steps and then gives the hand back to a given program.

Program transformation — In contrast with [22], we don't justify our machine by exhibiting a program transformation. We could give such a transformation, as it amounts to reveal the term behind the proof of the quantitative monoid part of Theorem 42.

6. Conclusion

We have proposed an abstract quantitative framework, built upon Krivine's classical realizability for system \mathbf{L}_{foc} and the notion of resource monoid developed in [18]. As a particular case of our construction, we have defined the quantitative reducibility candidates, which allow us to prove complexity properties of typable programs. Inside $\mathbf{MAL}\omega$, we then have defined a linear forcing interpretation of \mathbf{MAL} and an abstract machine that internalizes the computational behavior of the programs obtained through a particular instance of this forcing. We finally have proved a decomposition theorem which states that quantitative reducibility candidates for \mathbf{MAL} can be obtained as the composition of ordinary reducibility candidates and linear forcing.

We plan to explore several research directions.

Order sensitive realizability — Both classical realizability and quantitative classical realizability are insensitive to order, in the logical sense. Indeed, unlike reducibility candidates, these techniques are designed precisely to interpret second-order or higher-order logics. In [3], a resource sensitive realizability is defined. One particularity of this realizability is that it cannot be used to interpret second-order quantifiers (in the paper, only a *linear* second-order quantifier is interpreted), and thus allows an even finer grained study of the complexity properties of programs. This is achieved by using *typed* abstract bounds. It would be interesting to see if this framework and ours can be both generalized into a new one: it could lead to a even more precise quantitative analysis. Indeed, if we are able to study the complexity due to the presence of different exponentials, what about the complexity due to quantifiers?

Countdown and Implicit Complexity — Several attempts have been made to prove fundamental complexity results inside a purely logical framework. We can for example mention the work of Terui and al. [20] where a

link between focalization and space compression theorem is stated. In [1], the question is raised of whether it is possible to prove in their setting a hierarchy theorem like $\mathbf{P} \subsetneq \mathbf{EXP}$. It is striking that such an elementary complexity result cannot be easily proved in a proof theoretic setting. We conjecture that one of the main reasons for this apparent difficulty is the lack of expressivity of the logics at stake. For example, all known proofs of $\mathbf{P} \subsetneq \mathbf{EXP}$ crucially rely on defining a Turing machine which executes another Turing machine on an entry for a certain number of steps. This kind of feature is not available in a purely functional λ -calculus: it is not possible to internalize such a λ -evaluator inside the typed λ -calculus. Moreover, adding this feature would rise the problem of how to type programs using it. This is exactly the functionality implemented by the countdown machine. Using a variant of it, we would be able to execute programs for a certain number of steps. Typing those terms can be achieved using forcing. It seems to us we could use these facts to prove hierarchy theorems in a purely proof theoretic setting. As a first test for this idea, we plan to see whether it is possible to prove $\mathbf{P} \subsetneq \mathbf{EXP}$ in a forcing extension of the system described in [1].

Krivine Linear Algebras — We plan to reformulate this framework in a more abstract setting, in the spirit of Krivine’s realizability algebras [14]. In such a framework, we could express both quantitative realizability and linear forcing. We could hopefully prove a general iteration theorem of which our connection lemma would be a particular case.

A Logic of forcing — As already noticed, while the identification $\langle t | u \rangle \equiv \langle u | t \rangle$ made in \mathbf{L}_{foc} materializes the involutivity of negation, the identification $\langle u^\ominus | t \rangle \{K\} \equiv \langle t^\ominus | u \rangle \{K\}$ accounts for the properties of forcing orthogonality. This remark suggests a new logic where forcing orthogonality would be *primitive*, just like negation in linear logic. In such a setting, forcing would be easily recovered and dealt directly with. We are currently investigating a *logic of forcing*, which could be used as a type system for a calculus with effects.

Differential privacy and function sensitivity — *Differential privacy* [6] is a quantitative property of randomized functions (typically functions giving an answer to user queries on a database) that prevents malicious users to gain confidential knowledge from repeated queries. In [26], a linear type system that ensures differential privacy is proposed. It is based on *function sensitivity*, a measure of how the *distance* between outputs of a function is related to the distance between the respective inputs (this property is similar to *Lipschitz continuity*). We plan to see if the logical relations used in [26] to prove the soundness of their type system can be reformulated in terms of quantitative realizability.

References

- [1] P. Baillot. Elementary linear logic revisited for polynomial time and an exponential time hierarchy (extended version). *to appear in the Proceedings of Asian Symposium on Programming Languages and Systems (APLAS 2011)*, 2011.
- [2] P. Baillot and V. Mogbil. Soft lambda-calculus: a language for polynomial time computation. In *Foundations of software science and computation structures*, pages 27–41. Springer, 2004.
- [3] A. Brunel and K. Terui. Church \Rightarrow Scott = Ptime: an application of resource sensitive realizability. *Electronic Proceedings in Theoretical Computer Science*, 23, 2010.
- [4] P.J. Cohen. The independence of the continuum hypothesis. *Proceedings of the National Academy of Sciences of the United States of America*, 50(6):1143, 1963.
- [5] P-L. Curien and H. Herbelin. The duality of computation. In *ACM sigplan notices*, volume 35, pages 233–243. ACM, 2000.
- [6] C. Dwork. Differential privacy. *Automata, languages and programming*, pages 1–12, 2006.
- [7] J-Y. Girard. Linear logic. *Theoretical computer science*, 50(1):1–101, 1987.
- [8] J-Y. Girard. A new constructive logic: classic logic. *Mathematical Structures in Computer Science*, 1(03):255–296, 1991.
- [9] M. Hofmann. Safe recursion with higher types and bck-algebra. *Ann. Pure Appl. Logic*, 104(1-3):113–166, 2000.
- [10] M. Hofmann and P.J. Scott. Realizability models for bll-like languages. *Theoretical Computer Science*, 318(1-2):121 – 137, 2004. Implicit Computational Complexity.
- [11] S-C. Kleene. *Formalized recursive functionals and formalized realizability*, volume 89. Amer Mathematical Society, 1969.
- [12] J-L. Krivine. Realizability in classical logic. Course notes of a series of lectures given in the University of Marseille, may 2004 (last revision: july 2005). *Panoramas et syntheses, Société Mathématique de France*, 2005.
- [13] J-L. Krivine. A call-by-name lambda-calculus machine. *Higher-Order and Symbolic Computation*, 20(3):199–207, 2007.
- [14] J-L. Krivine. Realizability algebras: a program to well order R. *manuscript*, 2010.

- [15] J.-L. Krivine. Realizability algebras II: new models of ZF+ DC. *Arxiv preprint arXiv:1007.0825*, 2010.
- [16] U. Dal Lago and M. Hofmann. Bounded linear logic, revisited. In Pierre-Louis Curien, editor, *Typed Lambda Calculi and Applications*, volume 5608 of *Lecture Notes in Computer Science*, pages 80–94. 2009.
- [17] U. Dal Lago and M. Hofmann. A semantic proof of polytime soundness of light affine logic. *Theory of Computing Systems*, 46:673–689, 2010.
- [18] U. Dal Lago and M. Hofmann. Realizability models and implicit complexity. *Theoretical Computer Science*, 412(20):2029 – 2047, 2011. Girard’s Festschrift.
- [19] S. Lengrand and A. Miquel. Classical F [omega], orthogonality and symmetric candidates. *Annals of Pure and Applied Logic*, 153(1-3):3–20, 2008.
- [20] A. Saurin M. Basaldella and K. Terui. On the meaning of focalization. 6505:78–87, 2011.
- [21] A. Miquel. Classical program extraction in the calculus of constructions. In *Computer Science Logic*, pages 313–327. Springer, 2007.
- [22] A. Miquel. Forcing as a program transformation. In *Logic in Computer Science (LICS), 2011 26th Annual IEEE Symposium on*, pages 197–206. IEEE, 2011.
- [23] G. Munch-Maccagnoni. Focalisation and classical realisability. In *Computer Science Logic*, pages 409–423. Springer, 2009.
- [24] M. Okada. Phase semantic cut-elimination and normalization proofs of first-and higher-order linear logic. *Theoretical Computer Science*, 227(1-2):333–396, 1999.
- [25] M. Pagani and L. Tortora de Falco. Strong normalization property for second order linear logic. *Theoretical Computer Science*, 411(2):410–444, 2010.
- [26] J. Reed and B.C. Pierce. Distance makes the types grow stronger: A calculus for differential privacy. In *ACM SIGPLAN Notices*, volume 45, pages 157–168. ACM, 2010.